UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS FÍSICAS
Departamento de Física Aplicada III (Electricidad y Electrónica)

MODELACIÓN Y SIMULACIÓN ORIENTADA A OBJETO DE PLANTAS DE VAPOR

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Juan Antonio Gómez Pulido

Bajo la dirección del doctor

José María Girón Sierra

Madrid, 2002

ISBN: **978-84-669-0411-7**

UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS FISICAS
DEPARTAMENTO DE ELECTRONICA

MODELACION Y SIMULACION ORIENTADAS A OBJETO DE PLANTAS DE VAPOR

MEMORIA
presentada por

JUAN ANTONIO GOMEZ PULIDO

para optar al grado de Doctor en Ciencias Físicas

Director:
José María Girón Sierra

Indice:

Indice

MEMORIA

Introducción

Introducción

El tema abordado por la presente Tesis Doctoral ha sido la simulación de centrales térmicas de producción de electricidad. Nuestro objetivo ha sido abrir camino para realizar la simulación mediante Programación Orientada a Objeto, con especial atención a las posibilidades gráficas y de interacción con el usuario de las estaciones de trabajo.

Ciertamente la simulación tiene un papel cada vez más extenso e importante en el mundo industrial. La posibilidad de prever, y analizar de forma inocua, comportamientos de sistemas reales ofrece evidente interés en múltiples actividades (entrenamiento de operadores y pilotos, estudios ambientales y de epidemias, tráfico, manufactura, etc.). El avance de las técnicas numéricas y de las tecnologías de procesamiento digital, y el constante abaratamiento, trabajan en favor del uso de la simulación, incluso por sectores inesperados. Junto a este fenómeno de extensión y progreso, hay que contar también con el constante interés por la simulación, que mantienen, desde hace años, sectores fundamentales de la industria. Este es el caso, especialmente relevante, de la simulación de centrales productoras de energía eléctrica.

Puede distinguirse un doble uso de los simuladores de centrales: por una parte, como ayuda a la investigación y desarrollo, y por otra, como herramienta válida de enseñanza y entrenamiento. Atendiendo a estas posibilidades, y conscientes de su gran interés, nuestro Departamento de Informática y Automática, abordó la creación, en el ámbito universitario de nuestro país, de un simulador de cierta envergadura. Se aunaron esfuerzos de muchos de los miembros del Departamento a lo largo de cuatro años, haciendo honor a la tradición aplicada procedente del antiguo Departamento de Física Industrial. Se consiguió llevar a cabo con éxito, hasta la fase final operativa, un

simulador de central térmica mediante una tecnología, bien asentada en la realidad del sector eléctrico, que combinaba la programación en Fortran con la interactividad de un panel mímico de control a escala real.

La idea originaria de nuestra investigación surgió observando los resultados que acabamos de citar. Desde luego, un panel mímico se adecúa bien a la sistemática tradicional para entrenar operadores. Por otra parte, también parece útil el empleo de las posibilidades gráficas de las estaciones de trabajo. Una pantalla de ordenador tiene la ventaja de servir para muchas posibles visualizaciones, con entera libertad. De aquí parte la orientación básica de nuestra investigación: crear una simulación que saque partido de las virtualidades gráficas de las pantallas. En cuanto a la tecnología informática propia de las estaciones de trabajo, la vía actual preferida consiste en el empleo de X-Windows y el lenguaje C, o bien C++.

A la hora de plantear una simulación, existen diversos enfoques posibles, en cuanto a los fines perseguidos, la potencialidad correspondiente, y el grado de complejidad que se desea o puede afrontar. Desde un comienzo se ha tenido claro en nuestra investigación, que no se podía pretender un tipo de objetivos tan desmesurado, como para intentar igualar la labor de un nutrido equipo de investigadores durante cuatro años. Ahora bien, no cabe duda que, afortunadamente, el avance de la tecnología favorece el logro de resultados aceptables con menos personas y tiempo.

Alentados por esta última consideración, nos decidimos a afrontar la realización de una simulación básica aportando nuevas técnicas y con unas presentaciones visuales que expresen, de una manera más intuitiva, los fenómenos dinámicos más relevantes en las centrales térmicas.

La finalidad que perseguimos, en cuanto al uso de tal simulación, es lograr una base útil para los estudios y aprendizajes a nivel conceptual, sobre el ciclo termodinámico usado en las centrales y los subsistemas implicados, antes de entrar en los numerosos detalles técnicos de cada central en particular.

En cuanto a la tecnología, nuestro deseo ha sido situar nuestra simulación de centrales en el ámbito de las estaciones de trabajo (debido a su velocidad y posibilidades de adaptación al ambiente industrial) , de la Programación Orientada a Objeto, POO (por su especial adecuación a la modelación y simulación) y del entorno X-Windows (debido a su importancia como estandard, y a su impacto visual). Con ello, se consigue desarrollar un código transportable a un amplio conjunto de máquinas de uso tanto académico como profesional, bajo una normalización común.

Especialmente, cabe esperar del uso de POO diversas ventajas muy apreciadas hoy día, por quienes desarrollan grandes aplicaciones informáticas. En primer lugar, la claridad conceptual y la estructuración modular promovida por su metodología. En segundo lugar, la facilidad con que el código desarrollado para una aplicación, puede reutilizarse en otras (posiblemente semejantes, o que supongan una nueva generación). En nuestro caso, estas ventajas encierran una posibilidad prometedora: la de facilitar ulteriores creaciones de simulación, quizá especializada para alguna central concreta, a partir del código que hemos desarrollado.

Al revisar la breve historia de los denominados GUI, basados en ventanas y ratón, se encuentra que su origen es común con el de la POO, en el centro de investigación XEROX PARC. Este es uno de los motivos por el que la POO es connatural, casi exigida, para la creación de aplicaciones de gran impacto visual y manejo amigable, que es lo deseado al emplear X-Windows.

La exposición de nuestro trabajo comienza (Capítulo I) delimitando el marco de la investigación, y señalando cuáles son los objetivos que nos marcamos en vista a la situación actual del tema. Para ello, nos ayudaremos del estudio de la bibliografía más relevante, junto con los trabajos de investigación se han llevado a cabo en el contexto internacional, etc.

A continuación (Capítulo II), nos fijaremos en las bases tecnológicas de nuestro trabajo: estado actual y posibilidades de las herramientas de hardware y software.

A lo largo de la investigación hemos cubierto dos grandes etapas. La primera ha servido para introducirnos en los conocimientos y experiencias del caso, y la segunda, más extensa, para realizar la aportación clave.

Dedicamos el Capítulo III a exponer el trabajo realizado en la primera etapa. El objetivo que nos marcamos fué crear una simulación estática, que, por ejemplo, pudiera servir como herramienta de estudio a aquéllas personas que estén efectuando un primer contacto con el conocimiento y manejo de centrales térmicas. Teniendo en cuenta la buena adecuación de los PC para aplicaciones de uso personal, siempre que no requieran grandes prestaciones de hardware, decidimos emplear para esta etapa lenguaje C bajo entorno de Borland.

La parte más amplia de la Tesis se dedica a la segunda etapa de nuestro trabajo. La cuestión central es contemplar el comportamiento dinámico de los procesos: saber qué sucede a lo largo del tiempo, qué repercusiones pueden tener ciertos cambios en las variables, etc.

En el Capítulo IV exponemos un análisis inicial, de planteamiento. Principalmente nos han interesado tres aspectos: la evolución natural de las magnitudes termodinámicas del ciclo, la influencia del operario sobre el funcionamiento de la planta, y el control automático que convenga aplicar. Para tratar estos puntos hay diversas alternativas de modelación, que comentaremos en el Capítulo, señalando las decisiones que hemos adoptado para nuestro trabajo. Acto seguido, nos referimos a las fases fijadas para el desarrollo, a las técnicas y herramientas empleadas, el problema del coordinador ("scheduler"), y facetas de la POO y del uso de X-Windows.

Los dos siguientes capítulos se dedican a la modelación, y a la simulación.

A lo largo del Capítulo V se presentará un modelo teórico básico, según la metodología asociada al empleo de POO. Dicho modelo hace hincapié en los fenómenos termodinámicos fundamentales, obviando los aspectos técnicos de los dispositivos de una planta concreta. Consiste en un conjunto de componentes, que son, a su vez, modelos individuales de los principales subsistemas de una planta genérica. Diseñamos estos modelos de forma que puedan combinarse e interactuar, para reflejar la estructura y funcionalidad de diversas posibles configuraciones de centrales.

Es interesante considerar que la POO promueve un tipo de desarrollo basado en sucesivos prototipos. En efecto, es fácil mediante POO crear rápidamente unos objetos sencillos, que encapsulan los comportamientos más representativos de los subsistemas; y hacer que interactúen según la configuración de la planta que se desea simular. Se tiene así, relativamente pronto, un primer prototipo de modelo global más o menos acertado. Después viene una labor de refinamiento y enriquecimiento conductual de los objetos, para obtener un prototipo más válido y completo. Siempre es posible modificar el contenido de un objeto (variables y comportamientos), y ver las consecuencias derivadas al interactuar con otros objetos. Precisamente esta plasticidad es muy interesante para nuestro caso: porque facilita la adaptación del modelo de carácter genérico desarrollado, a casos específicos de centrales concretas.

En general, el problema más difícil de programación mediante POO se da justamente al comienzo: hay que pensar muy bien la estructuración en objetos. Se recomienda que la jerarquía diseñada tenga pocos niveles, porque los mecanismos de herencia repercuten en más tiempo de proceso. Esta es una razón importante para centrar nuestro trabajo en un modelo conceptual, para que sirva como referencia y base estructural de objetos: pasar a un mayor nivel de detalle técnico supondrá requerimientos más fuertes de velocidad de proceso.

En el Capítulo VI, expondremos la simulación que hemos desarrollado, haciendo uso del modelo para los fines que hemos previsto. Daremos numerosos detalles de la realización, con especial mención de los objetos, el tipo de control que se efectúa, y la actuación del coordinador.

Finalmente, resumiremos en forma de conclusiones los resultados obtenidos, incluyendo un escueto análisis del alcance, límites, y aplicabilidad de lo aportado por nuestra investigación.

Adjuntamos unos apéndices con la bibliografía consultada, y listados de los programas que hemos realizado.

Deseamos haber contribuído con nuestro trabajo a la difícil tarea asociada al diseño y manejo de centrales térmicas, teniendo a la vista las preocupaciones actuales respecto a la generación de energía, que parecen reclamar renovaciones de concepción y soporte en este importante sector de la industria.

Queremos expresar aquí nuestro agradecimiento al Profesor D. José María Girón Sierra por su labor de dirección y supervisión, y por la aportación de material técnico y bibliográfico, sin el cual no hubiera sido posible alcanzar los resultados obtenidos. Agradecemos también al resto del equipo de Control Automático de nuestro Departamento por su colaboración y ayuda para que las investigaciones se hayan desarrollado a un buen ritmo. Finalmente, agradecemos a cuantas personas, dentro y fuera del Departamento, han contribuido de diversas formas a la feliz consecución de la Tesis.

Capítulo I:

MARCO DE LA INVESTIGACION

Capítulo I:

MARCO DE LA INVESTIGACION

Dedicamos este Capítulo a encuadrar nuestro tema de investigación dentro del área en que se produce, con el propósito de establecer aspectos en que es posible introducir innovaciones y realizar aportaciones de interés.

Dividiremos el Capítulo en tres apartados.

En el primer apartado, daremos una panorámica del problema de la modelación y simulación, centrando la atención en las plantas térmicas de vapor para la producción de energía eléctrica.

El segundo apartado nos lleva a sentar las bases de nuestro trabajo. Delimitamos, de forma razonada, los objetivos y planteamiento de la modelación y de la simulación a realizar, señalando pasos a seguir y metas prácticas. Hacemos una breve evaluación de posibilidades de originalidad, y de dificultades asociadas.

Finalmente, en el tercer apartado revisamos el material bibliográfico, para dar una idea del estado de la cuestión, las direcciones en que se mueve la investigación, y las posibles carencias a las que dirigir nuestro interés. Parte de la documentación recogida ha sido de gran importancia para nuestro trabajo, como fuente de información conceptual y técnica.

I.1.- PANORAMICA.-

I.1.1.- Modelación, Simulación, y Simuladores.-

"Simular es utilizar modelos con un fin científico o profesional". Así podíamos resumir en una sola frase lo que es la base de nuestro trabajo. Para comprender los fines que persigue la simulación, podemos decir que hay dos campos de actuación o interés: por un lado lo analítico y por otro lo formativo. De esta forma, cabe encontrar tres tipos de simuladores, dedicados a:

- **Análisis.** Son los que cumplen la faceta más científica de la simulación. Estudian comportamientos estáticos y/o dinámicos de los sistemas. Son muy usados para el estudio de los fenómenos que se producen en la realidad, analizándolos para conseguir predicciones, elaboración de estadísticas, reconocimiento de sucesos por los síntomas, detección de averías, etc.

- **Entrenamiento y/o enseñanza.** Son conocidos normalmente como Simuladores ("Simulators"). Pueden tener tanto una vertiente didáctica (enseñar comportamientos de los sistemas para unos pretendidos fines, comprender los fenómenos físicos que se producen, mostrar características de funcionamiento, etc.), como de entrenamiento (formar a un operador en el manejo de controles, en la respuesta a alarmas, en los modos de operación normal y con incidencias, etc.).

- **Híbridos de los dos anteriores.** Destacan por su complejidad. Suelen ofrecer varias alternativas de operación para poder analizar habiendo previamente aprendido.

Dentro de las realizaciones de simulación se observa una diversidad de niveles de detalle. Hay simuladores "full-scope", que abordan toda la problemática del sistema. Hay simuladores más simplificados, que llegan incluso hasta considerar sólo los principios más básicos. Hay simuladores que se fijan en subsistemas concretos. Todo ello queda reflejado en la bibliografía científica y técnica como una gran variedad de publicaciones, que obedecen a distintas necesidades y enfoques.

Se denominan simuladores a dispositivos (quizá dotados de un panel de controles electromecánicos, o bien puramente informáticos) que efectúan procesos de simulación. Previamente, para llevar a cabo dichos procesos, es necesario disponer de un modelo del sistema objeto de simulación. La forma de modelar sistemas depende del fin que se persiga. Hay modelos que contemplan sólo aspectos estáticos, otros (la mayoría) entran a considerar los dinámicos. Desde el punto de vista matemático, existen diferentes alternativas en cuanto a técnicas y precisión.

I.1.2.- La Simulación de Centrales Térmicas.-

Según hemos mencionado, el campo de intereses de la simulación es muy amplio y tiende a expandirse. Son conocidos los simuladores de vuelo, de comportamientos del mercado en bolsa, de diseño electrónico, de estrategia militar, de meteorología, etc. A nosotros nos ha interesado concretamente la problemática de las centrales térmicas para la producción de energía eléctrica. Como tales pueden considerarse tanto las nucleares, como las de combustión de carbón (u otros materiales). Nuestro trabajo se relacionará más con estas últimas. Sin embargo, ambas tienen numerosos puntos en común (elementos concretos, técnicas de simulación, etc).

Hagamos una revisión de lo que nos ofrece la bibliografía respecto a características, tendencias y posibilidades en el sector de la investigación que nos interesa. Las figuras 1.1, y 1.2, presentan un resumen esquemático de los trabajos que nos han parecido más representativos.

Por supuesto, no tratamos de ser exhaustivos, ni tampoco pretendemos llegar a cuestiones de matices; porque no está a nuestro alcance, siendo de por sí todo un tema de investigación documental. Para dar una idea de la magnitud de la masa bibliográfica disponible en cuanto a generación de energía eléctrica, baste considerar que la guía E.P.R.I. (Electric Power Research Institute, el centro de coordinación de investigaciones sobre energía más importante en U.S.A.), correspondiente al período 1982-1987, y que obra en nuestro poder, es una especie de listín telefónico que clasifica sus proyectos de investigación mediante unas 2500 palabras clave. Nos parece muy informativo subrayar algunas de las claves, y el número de trabajos (reports) de investigación correspondiente:

— "Simulation", ... 25 reports.

— "Simulators", ... 7.

— "Computer Simulation", . . . 56.

— "Reliability", ...170.

— "Fossil-Fuel Power Plants", ... 96.

— "Power Plant Dynamics", ... 6.

— "Thermodynamic Cycles", ... 3.

— "Models", ... 7.

— "MMS Code", ...11.

— "Mathematical Models", ...55.

— "Process Control", ... 7.

— "Computer Control Systems", . . .17.

— "Failures", ...32.

— "Pilot Plants", ...22.

— "Design", ...92.

— "Technology Assessment", ...75.

— "Tubes", ...56.

— "Turbines", ...27.

— "Boilers", ...34.

Por otra parte, el catálogo de programas de ordenador desarrollados bajo los auspicios del E.P.R.I., recoge un total de 156 en diversas categorías. De ellos, sólo 2 son simulaciones de planta completa.

a) Modelación.-

En las técnicas de modelación de las plantas hay variedad de criterios. Los modelos se basan, desde puras abstracciones matemáticas a partir de la teoría de los fenómenos físicos ([Li-Chi, 1986], [Kwatny, 1986]), hasta la cristalización en curvas y ecuaciones de una masa ingente de informaciones obtenida tomando datos experimentales de una central ([Zwingelstein, 1986], [Bernard, 1986]), pasando por una modelación teórica ajustada después a unos datos experimentales ([Chaplin, 1986], [Ocampo, 1984]). Para ciertos intereses, la situación ideal consiste en disponer de una planta real, con libre acceso a ella en cuanto a poder experimentar para recoger datos, y con ellos elaborar curvas, identificar sistemas,etc.; los modelos así creados serán muy fiables, pero limitados a esa planta con esos dispositivos concretos, y bajo unas condiciones de operación definidas, lo cual restringe considerablemente la aplicabililidad del modelo a otros casos. Por otra parte, el estudio de los fenómenos físicos que se producen no exige necesariamente el acceso a una planta; si bien su complejidad y dificultad se acentúa a medida que se intente considerar los más ínfimos detalles, para depurar el modelo. La ventaja de un modelo basado en la teoría y no en la experimentación es que puede ser usado para simular bajo todo tipo de circunstancias, (cosa que la experimentación directa no permite), ampliando así el rango de aplicabilidad. Pero la desventaja es que esta simulación tiende a estar lejos de la realidad, debido a la dificultad de contemplar todos los aspectos concretos de una central. Entre ambas posturas de modelación, hay toda una sucesión de estados intermedios que combinan la teoría con la experimentación.

> **El problema del agua.** Un ejemplo ilustrativo del hecho de abordar multitud de aspectos en la elaboración del modelo es cuando consideramos que, normalmente, el fluido de trabajo que describe el ciclo termodinámico de la planta es el agua. El uso del agua significa encontrarse con una serie de fenómenos principales y muchos otros secundarios. No se puede afirmar que se trata de una componente lineal cuya caracterización quepa fácilmente en ecuaciones sencillas; de hecho, habremos de apoyarnos sobre tablas distintas para los distintos estados posibles del agua. Y además, hasta tal punto es importante este asunto, que hay una gran cantidad de investigaciones que se centran en fenómenos secundarios, como las formas de las gotas del agua, los efectos de movimiento en los dispositivos, etc.

Otro aspecto de la modelación es bajo qué representación se plantea: variables de estado temporales, o variables lineales en el dominio de la transformada de Laplace (s) ([Azuma, 1975], [Kubiak, 1984]). Este último caso es particularmente importante cuando se persiguen los aspectos de control del sistema. Normalmente, todos parten de ecuaciones temporales en variables de estado, y los que están interesados en el control, efectúan las simplificaciones necesarias para linealizar las ecuaciones y

representar el modelo mediante funciones de transferencia en "s", de forma que puedan aplicarse las técnicas y conocimientos del control de procesos. Sin embargo, el hecho de utilizar variables de estado temporales no significa renunciar al control ([Sukanit, 1986], [Chaplin, 1986], [Bernard,1986], [Rajakuman, 1986], [Usoro, 1977], [Masada, 1979]).

Una tendencia reciente es la de buscar conocimientos en profundidad de aspectos específicos de las plantas, de forma que son menos los trabajos que consideran a la planta en su totalidad como el objetivo de la modelación ([Riley, 1984]). Ahora se simulan elementos concretos (turbinas, chimeneas, calderas, etc) con el fin de aumentar la seguridad, el rendimiento, etc., y también para poder posteriormente simular configuraciones distintas de plantas, integrando sus elementos. Así, hay simulaciones de elementos, preparadas para integrarse posteriormente en la simulación de una planta ([Kanodia, 1988], [Stegeman, 1986], [Belblidia, 1986], [Ocampo, 1984], [Masada, 1979], [Usoro, 1977], etc.); y otras de elementos aislados considerados en su individualidad ([Li-Chi, 1986], [Babula, 1986], [Kwatny, 1986], [Sukanit, 1986], [Chaplin, 1986], [Kubiak, 1984], [Roldán-Villasana, 1984], etc).

Existen trabajos de modelación, focalizados en considerar estáticamente en el tiempo magnitudes de interés, según un determinado punto de operación ([Riley, 1984], [Dunne, 1984]). Estos modelos, poco habituales en la literatura, guardan menos interés que los modelos dinámicos; porque, entre otras cosas, un buen modelo dinámico también llega a mostrar el comportamiento estático, tras un intervalo de tiempo y unas actuaciones adecuadas; además, en el plano didáctico, de entrenamiento, o en el de análisis, gran parte de la atención se dirige a ver qué controles han de actuar y qué valores iniciales han de darse para alcanzar unos valores estacionarios concretos que nos aseguren, por ejemplo, un buen rendimiento.

En resumen, la modelación puede considerar desde el estado estacionario hasta el dinámico; desde modelar el conjunto de la planta hasta hacerlo con los elementos individuales; desde concentrar la atención sólo en el desarrollo natural del proceso, hasta considerar los aspectos de control; desde partir de un estudio en variables de estado, hasta efectuarlo exclusivamente en el dominio de la transformada de Laplace. Combinando en mayor o menor grado estos elementos, aparece un amplio abanico de concepciones a la hora de modelar una central.

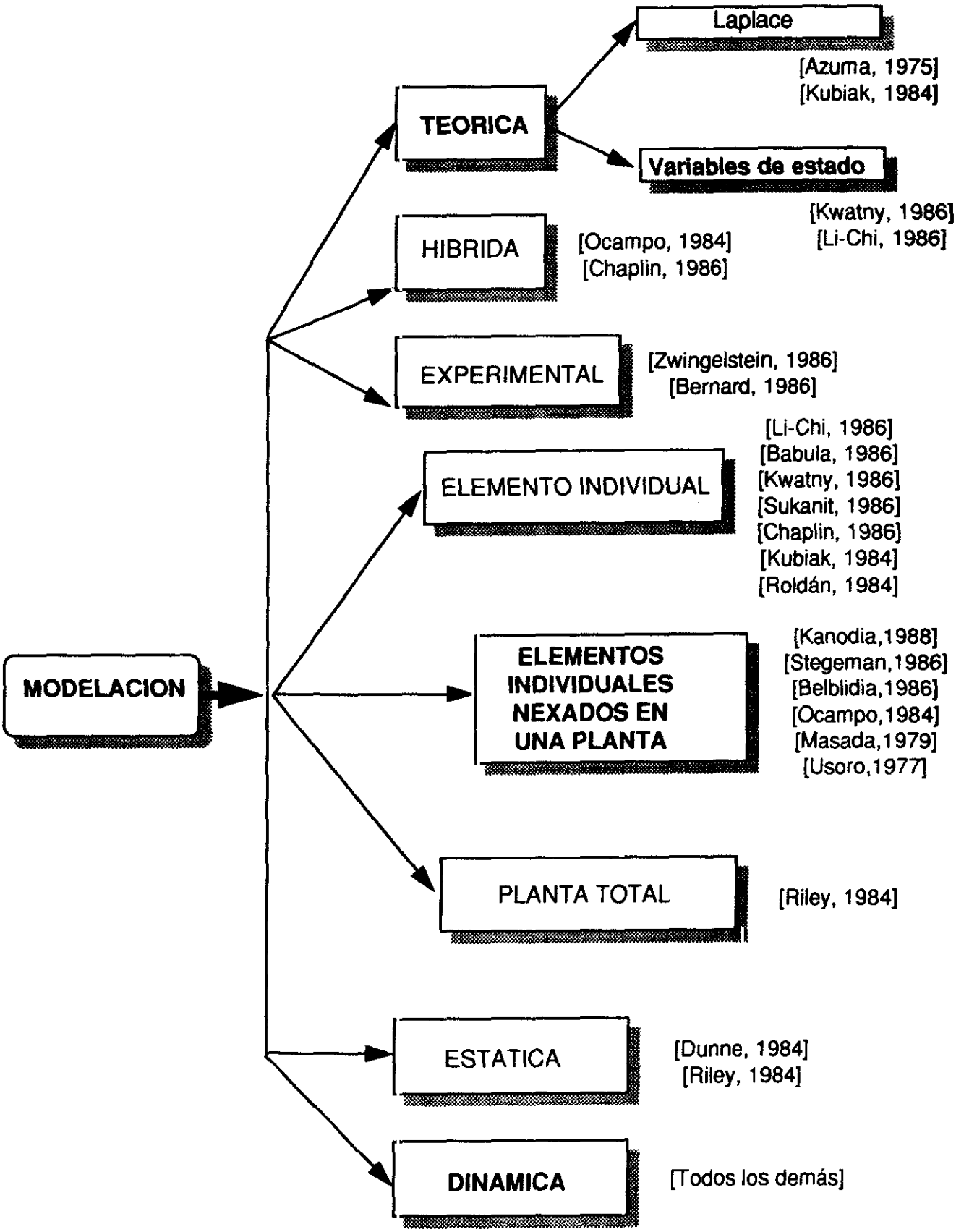

MODELACION
TEORICA
Laplace
[Azuma, 1975]
[Kubiak, 1984]
Variables de estado
[Kwatny, 1986]
[Li-Chi, 1986]
HIBRIDA
[Ocampo, 1984]
[Chaplin, 1986]
EXPERIMENTAL
[Zwingelstein, 1986]
[Bernard, 1986]
ELEMENTO INDIVIDUAL
[Li-Chi, 1986]
[Babula, 1986]
[Kwatny, 1986]
[Sukanit, 1986]
[Chaplin, 1986]
[Kubiak, 1984]
[Roldán, 1984]
ELEMENTOS INDIVIDUALES NEXADOS EN UNA PLANTA
[Kanodia,1988]
[Stegeman,1986]
[Belblidia,1986]
[Ocampo,1984]
[Masada,1979]
[Usoro,1977]
PLANTA TOTAL
[Riley, 1984]
ESTATICA
[Dunne, 1984]
[Riley, 1984]
DINAMICA
[Todos los demás]

Figura 1.1

b) Simulación.-

La aspiración de los trabajos científicos es moverse en un plano de ideas que sean útiles para bastantes personas. Esto implica situarse en un cierto grado de abstracción. En este sentido, a todos nos es familiar, en el ámbito académico, el empleo de diagramas de bloques y el de técnicas generales que en cierta forma permiten aislarnos de los problemas concretos. Por su parte, el ingeniero al que le encargan la realización de un proyecto, ve como problema más importante el llevarlo a un cumplimiento operacional, y si llega a publicar los resultados, éstos suelen tener el atractivo de ver plasmados de modo útil y práctico los principios científicos. En tales trabajos siempre hay un aspecto importante: los problemas más interesantes y muchas veces más arduos, son los de la vida real. Entre los extremos marcados por el tratamiento abstracto, y el técnico concreto, aparece un amplio espectro de publicaciones en torno a la simulación.

Un primer criterio para discriminar los tipos de aportaciones contenidas en la bibliografía, se refiere a dos polos de interés: simulaciones destinadas al puro análisis técnico de las plantas ([Zwingelstein, 1986], [Bernard, 1986]), identificación mediante la toma de datos, simulación para la optimización, etc.; y simulaciones orientadas al entrenamiento y enseñanza ([Kanodia, 1988], [Li-Chi, 1986], etc.). Entre ambas concepciones hay diversas posiciones intermedias, acentuando más o menos los aspectos didácticos o los analíticos, dependiendo del fin perseguido.

La simulación puede hacerse basándose en unos modelos propios, o bien sobre modelos ajenos implementados en lenguajes comerciales, como el MMS ([Barcelo, 1985], [Smith, 1985], [Rosard, 1985]), del cual hablaremos un poco más extensamente en el siguiente capítulo.

Cabe decir, respecto a la disponibilidad de un buen modelo, que tener acceso a una central térmica real, para su estudio, supone una especie de privilegio, al poder obtener una enorme masa de datos experimentales; este privilegio significa al mismo tiempo abordar un extenso trabajo de análisis. A este nivel se mueven casi exclusivamente las empresas especializadas. En algunas universidades (M.I.T.) se ha llegado a construir plantas piloto de cierta entidad para poder así disponer en el ámbito académico de una apropiada referencia experimental ([Bernard, 1986]). Precisamente, han sido muy importantes para nuestra tarea, varias Tesis Doctorales, de reconocido prestigio, que hemos tenido ocasión de obtener en el mismo M.I.T. ([Usoro, 1977], [Masada, 1979], [Azuma, 1975]); a través de ellas tenemos una cierta referencia real para nuestro trabajo, pues fueron desarrolladas para estudiar aspectos de plantas experimentales.

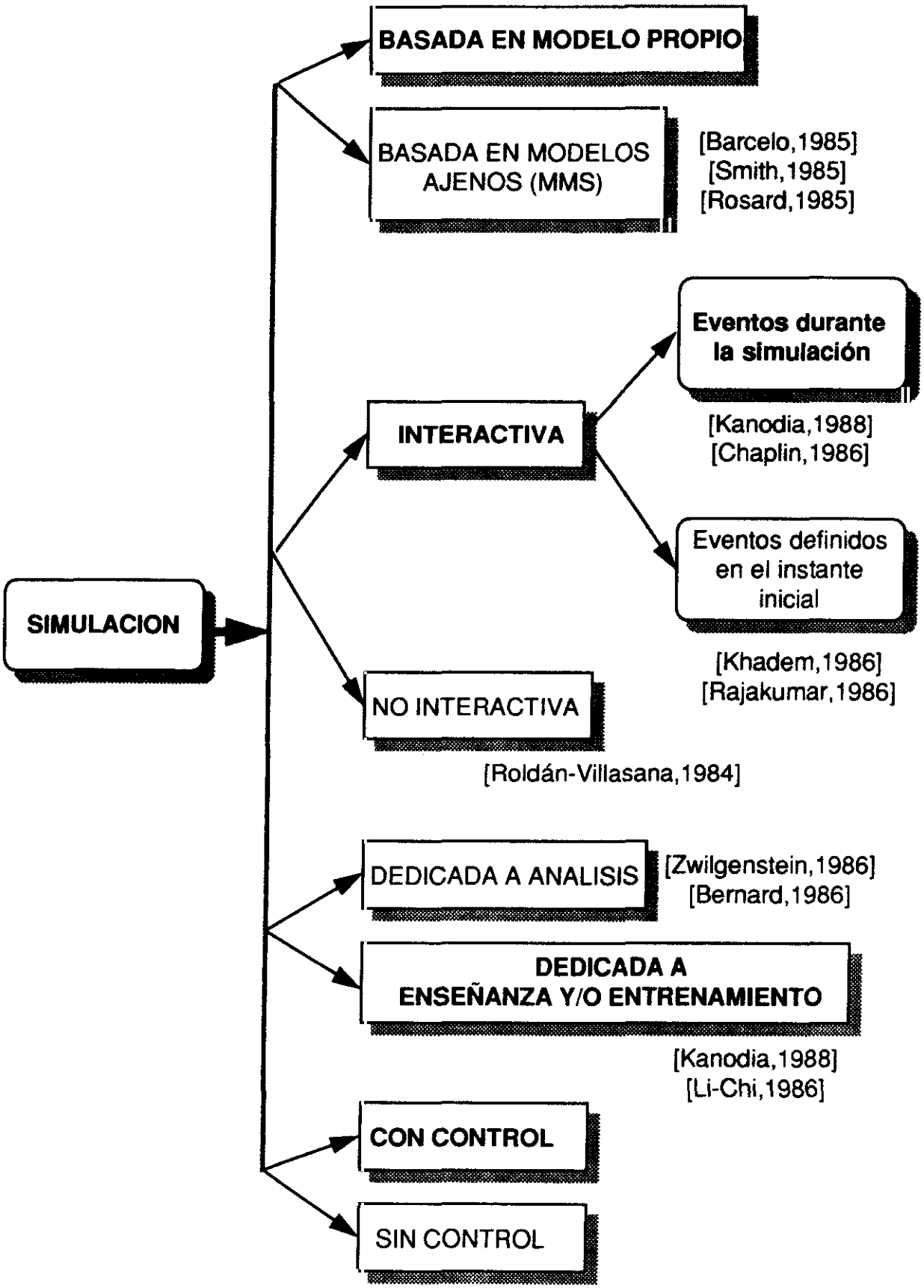

SIMULACION
BASADA EN MODELO PROPIO
BASADA EN MODELOS AJENOS (MMS)
[Barcelo,1985]
[Smith,1985]
[Rosard,1985]
INTERACTIVA
Eventos durante la simulación
[Kanodia,1988]
[Chaplin,1986]
Eventos definidos en el instante inicial
[Khadem,1986]
[Rajakumar,1986]
NO INTERACTIVA
[Roldán-Villasana,1984]
DEDICADA A ANALISIS
[Zwilgenstein,1986]
[Bernard,1986]
DEDICADA A ENSEÑANZA Y/O ENTRENAMIENTO
[Kanodia,1988]
[Li-Chi,1986]
CON CONTROL
SIN CONTROL

Figura 1.2

También, hemos de resaltar que muchos de los simuladores se ven provistos de la capacidad de interactuar en tiempo de simulación ([Kanodia, 1988], [Chaplin, 1986]), o antes de que comience la simulación ([Khadem, 1986], [Rajakumar, 1986]), con el objeto de analizar los comportamientos ante unas determinadas incidencias. De esta manera, es posible estudiar posibles averías, estados anómalos, etc. Otros trabajos no consideran esta interactividad, limitándose a simular el comportamiento natural del modelo para unas condiciones iniciales dadas ([Roldán-Villasana, 1984]).

Dicha interactividad suele atender a aspectos de control manual. Por ejemplo, puede ser el caso de cerrar la válvula que gobierna el aporte de gas al elemento de combustión, para controlar la temperatura de la caldera. La actuación correspondiente a la acción manual en la planta, puede hacerse en la simulación mediante teclado y/o ratón , o mediante los botones y otros accionadores en un panel mímico.

Además de controles manuales, existen lazos de control automático. Esto se incluye en la simulación, a través del software, bien por una formulación de funciones de transferencia, bien por variables de estado, dependiendo de la forma en que se elaboró el modelo.

c) Simuladores.-

Si se aborda la simulación de una central con un nivel de detalle grande, puede terminarse con posibles problemas numéricos. La experiencia mencionada en los Congresos por parte de las empresas dedicadas a los simuladores "full-scope", dice que si se quiere una simulación en tiempo real o más acelerado, se impone bien el uso de un ordenador grande, veloz (cuyo coste puede ser prohibitivo), o bien simplificar los modelos. Tenemos así un compromiso entre precisión y sentido económico (cabe preguntarse acerca de la utilidad de una gran precisión para objetivos de entrenamiento a un primer nivel).

La importancia actual de la simulación de estos sistemas es que, en nuestra opinión personal, todo llama a la creación de una nueva generación de centrales eléctricas. Esto parece cierto para el caso de las centrales nucleares: ya hay abundante parque de centrales anticuadas, no exentas de peligros, buscándose respuestas en conexión con la seguridad. Es claro que la creación de una nueva generación de centrales supone una macroinversión difícil de realizar. Hasta entonces, parece conveniente ir desarrollando simuladores de tecnología avanzada, en consonancia con el progreso tecnológico de nuestro tiempo.

A continuación nos proponemos dar una visión panorámica sobre tipos y características de simuladores dedicados a plantas térmicas. Hemos podido encontrar

abundante información técnica, de la cual desecharemos las destinadas a plantas nucleares por no ser éste nuestro propósito (aunque las características de simulación, requisitos, entornos, etc. sí pueden ser comunes), y de las restantes haremos una selección de casos representativos temáticamente, pues son numerosas las empresas en el mundo destinadas a esta actividad.

- **COMPACT PROCESS SIMULATOR**, de la empresa sueca **STUDVISK AB** es un simulador de entrenamiento full-scope de planta térmica, que contempla desde emergencias hasta análisis y predicción, y se basa en un panel mímico ayudado de monitores para visualizar gráficas.

- **WESPORT** (Westinghouse Engineering Simulator for Personalized Operating Real-Time Training) de **Westinghouse**, es un simulador destinado a estudiantes y operarios de plantas térmicas, desarrollado en 1987. El usuario puede, en tiempo de simulación, controlar la planta. Tras unas condiciones iniciales introducidas, se estudia la operación en estado estacionario, los transitorios de arranque y parada, la dinámica de las averías de los diversos elementos, los estados anómalos de operación, etc. Está implementado en un panel mímico ayudado por monitores.

- Un concepto distinto de simulador es **PROTEUS PC** (Program for Optimal Thermal Energy Utilization in Systems), de Gilbert International (USA), pues se trata de un producto específico para PC's destinado a la enseñanza y entrenamiento de plantas térmicas. Es un programa que modela y simula el balance másico y energético del sistema. Posee una librería de componentes (turbinas, bombas, compresores, calderas, intercambiadores de calor, condensadores, válvulas, etc.) que le permiten simular configuraciones distintas de plantas. Trae dos librerías más: una de propiedades, que implementan las tablas de vapor con los distintos estados del agua y otros gases, y una librería de control que facilita la manipulación de datos y la modelación del sistema de control. Cualquier PC, XT ó AT compatible, con coprocesador matemático, puede servir de soporte. Pese a su apariencia modesta, es completo y, además, caro (3000 $).

- La empresa australiana **COSTEC** ha desarrollado tres conceptos de simuladores:

  — Simuladores Compactos (50.000 $ del año 88): Ideales para los principios básicos de entrenamiento del operario. Con un panel mímico opcional para controles e indicadores, intenta ser una opción barata al integrar en una sola unidad (teclado y consola de tres monitores) todo lo que es el simulador. Así, se desarrollaron los simuladores FFPS (Complete Fossil-Fired Power Station), COGEN (Combined Cycle Gas Turbine, Heat Recovery Boiler), y otros.

  — Simuladores Genéricos (500.000 $): Ideales para el análisis de procesos, optimización del control y entrenamiento de operarios para planatas térmicas. Se basan en un panel simplificado y ofrecen el 75% de un simulador *full-scope*.

  — Simuladores Full-Scope (2 millones $): Reproducen el comportamiento de una planta térmica particular, analizando procesos, optimizando procesos, con un concepto de entrenamiento avanzado.

- La empresa francesa **CORYS** ha desarrollado, entre otros, un simulador de planta térmica para el entrenamiento en condiciones de operación normales, accidentes y control. Se basa más en paneles mímicos que en monitorización por consolas. Una estación instructora basada en una mini-computadora de 32 bits domina los pupitres mímicos, llevando a cabo una serie de funciones simuladas en tiempo más o menos acelerado a voluntad, tras haber definido unas condiciones iniciales.

- **ABB Power Automation** añade a sus simuladores de planta térmica una extensa librería de modelos de plantas con numerosos modos de operación. Concebidos para entrenamiento, dispone de gráficos interactivos, simulación de la planta entera o de elementos individuales, consolas para el instructor y para el ingeniero. Dos modos de simulación: por panel mímico de control o por computadora, etc.

- **Argus Technological Systems** desarrolla sus simuladores para el entrenamiento de operarios. Dispone de simulación en tiempo real y diverso software relacionado.

- **Atlantic Simulation Inc**. también desarrolla simuladores de plantas de vapor en general o para tareas concretas.

- **Bharat Heavy Electrical Ltd.**, *de la India, tiene todo tipo de simuladores* de planta, desde el más sencillo hasta "full-scope". **Macmet India Pvt. Ltd.**, también de esta nacionalidad, realiza su simulador de entrenamiento para que pueda soportarse en ordenadores PC.

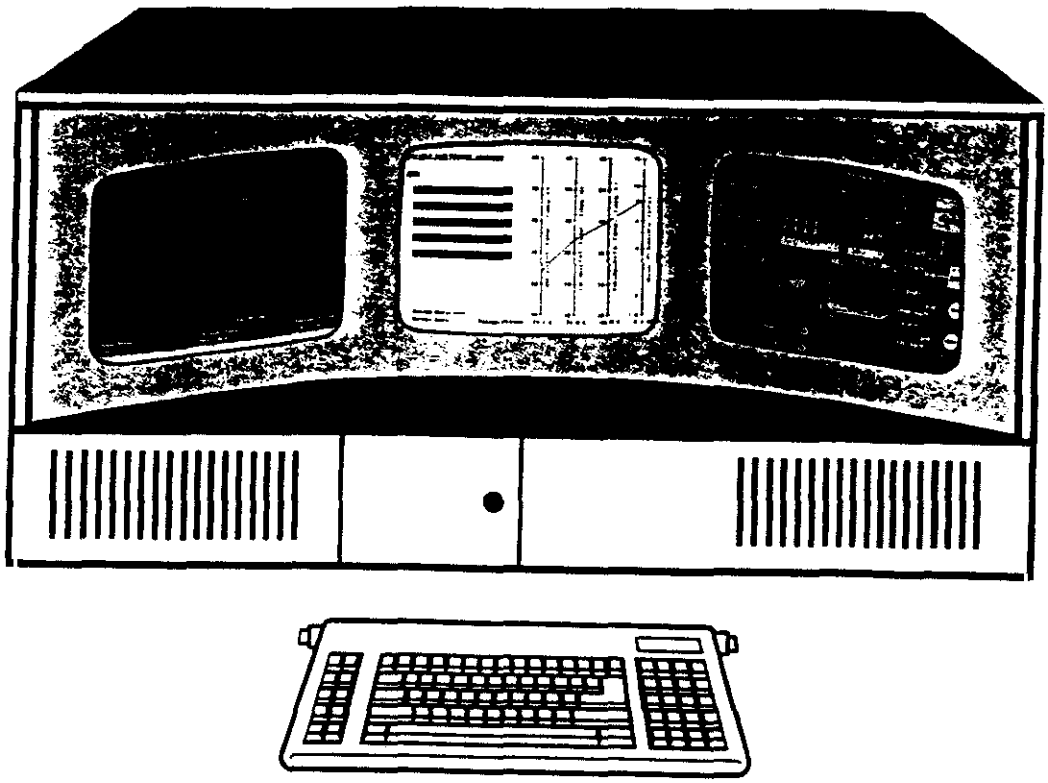
- La empresa canadiense **CAE Electronics Ltd.** concibe su simulador para "full-scope" ó tareas parciales. Tiene un modo de testeo en "background" (es decir, dejando libre al sistema para otras tareas mientras hace la simulación), con objeto de contribuir al mantenimiento de la planta.

- Las empresas **Computer Simulation Corp.** y **General Physics Corp.**, hacen proyectos de estos simuladores, software de simulación, etc.

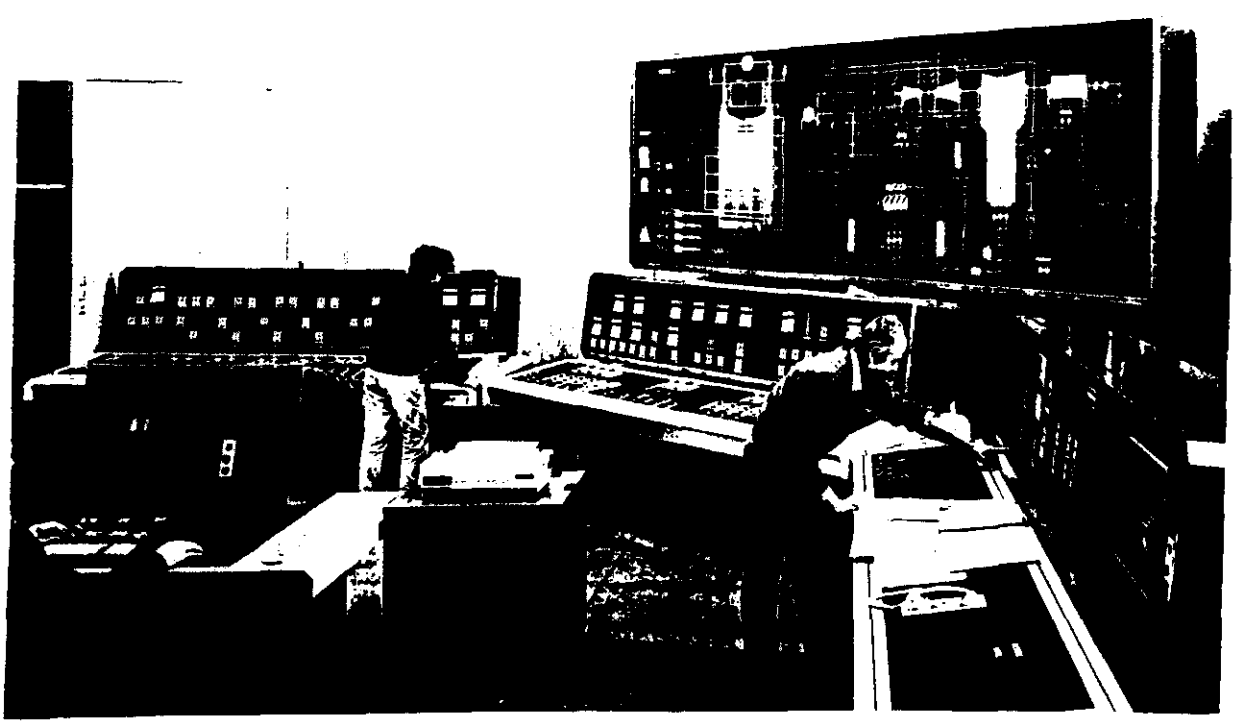
- **Management Analysis Company** ofrece para sus simuladores los sistemas expertos. Se basan tanto en paneles mímicos como en ordenadores, haciendo hincapié en la simulación para tareas concretas.

- **Marconi Simulation** dedica atención al control de procesos y a la ingeniería, pudiendo catalogarse sus simuladores como "full-scope" dedicados al análisis y desarrollo.

- Por último, citamos a **Nokia Information Systems**, que ofrece la novedad de interactuar con la simulación mediante pantallas táctiles. También ofrece software de simulación.



Compact Simulator



Full scope simulator

I.2.- PLANTEAMIENTO DE LA INVESTIGACION.-

A partir de los aspectos discutidos en el apartado anterior, y que resumen las Figuras 1.1 y 1.2, nos disponemos a definir objetivos, y a plantear las fases de nuestra investigación, pensando también en las herramientas necesarias.

I.2.1.- Definición de Objetivos.-

El eje conductor de nuestra investigación consistirá en considerar la planta industrial de vapor como sistema que efectúa un ciclo termodinámico.

A este respecto la bibliografía consultada, muestra diversos estudios de gran calidad, realizados para dar respuesta a aspectos concretos. Algunos modelan dispositivos individuales, otros utilizan librerías MMS para editar plantas, otros concentran más su atención en monitorizar comportamientos con vistas a la enseñanza, otros dan respuesta al problema del control, etc.

Frente a lo ya hecho, según recoge la literatura, deseamos aportar la integración de una multiplicidad de aspectos para dar una solución única, homogénea. Esto sin pretender el nivel de complejidad de ciertos simuladores profesionales (que requieren grandes instalaciones informáticas). Si bien, ofreciendo características interesantes, derivadas de los nuevos métodos y tecnologías que empleamos.

Centramos la investigación en un prototipo de planta de vapor. Los mayores esfuerzos deben dirigirse al ciclo termodinámico del aguas en dicha planta. Podemos ennumerar los puntos de interés así:

- Cómo se comporta el ciclo en el estado transitorio.
- Cómo responde ante sucesos en el tiempo.
- Cómo monitorizar y manipular durante la simulación.

- Cómo elegir los parámetros adecuados y en qué cuantía para obtener un mejor rendimiento u otra característica adecuada.

- Cómo guardar y recuperar simulaciones.

- etc.

En la Figura 1.3 podemos ver esquematizada la idea que perseguimos. Deseamos responder a una falta de visión global, observable en la literatura, que deriva de centrar la atención más sobre unos aspectos y menos sobre otros. Así, [Kubiak, 1984] y [Azuma, 1975] tratan a fondo la modelación lineal para el control automático, pero abandonando más los aspectos propios de la simulación. [Kanodia, 1988] busca integrar elementos en conFiguraciones distintas de plantas, pero renunciando a una adecuada monitorización; por contra, [Khadem, 1986] visualiza muy bien, pero sin interactividad en tiempo de simulación; etc. Nos parece importante que las soluciones individuales a problemas concretos puedan integrarse en una sola solución, alcanzando así un nivel de aplicabilidad más alto. Estas soluciones integradas no son patrimonio de los grandes simuladores "full-scope", sino que mediante trabajos incluso muy modestos, se puede tratar una buena cantidad de cuestiones. Debe haber una proporcionalidad razonable entre los medios aplicados, y los resultados obtenidos, respecto a las motivaciones concretas de cada trabajo.

Los avances de la tecnología suponen un constante reto, en cuanto a saber aprovechar de la mejor manera sus nuevas posibilidades. Nuestra intención es desarrollar una aplicación basada en tecnologías relativamente novedosas, en el contexto de la simulación de centrales; que tenga utilidad en las etapas de estudio, entrenamiento, o análisis, cuya clave sea la comprensión de los fenómenos fundamentales; y que pueda servir de referencia para sucesivas generaciones, quizá más especializadas, para valorar requerimientos de hardware y software.

Para desarrollar nuestra simulación, consideraremos un prototipo de central, y haremos un modelo en el que, efectuando simplificaciones razonables, y teniendo en cuenta ciertos límites, consigamos integrar la mayor cantidad de aspectos posibles.

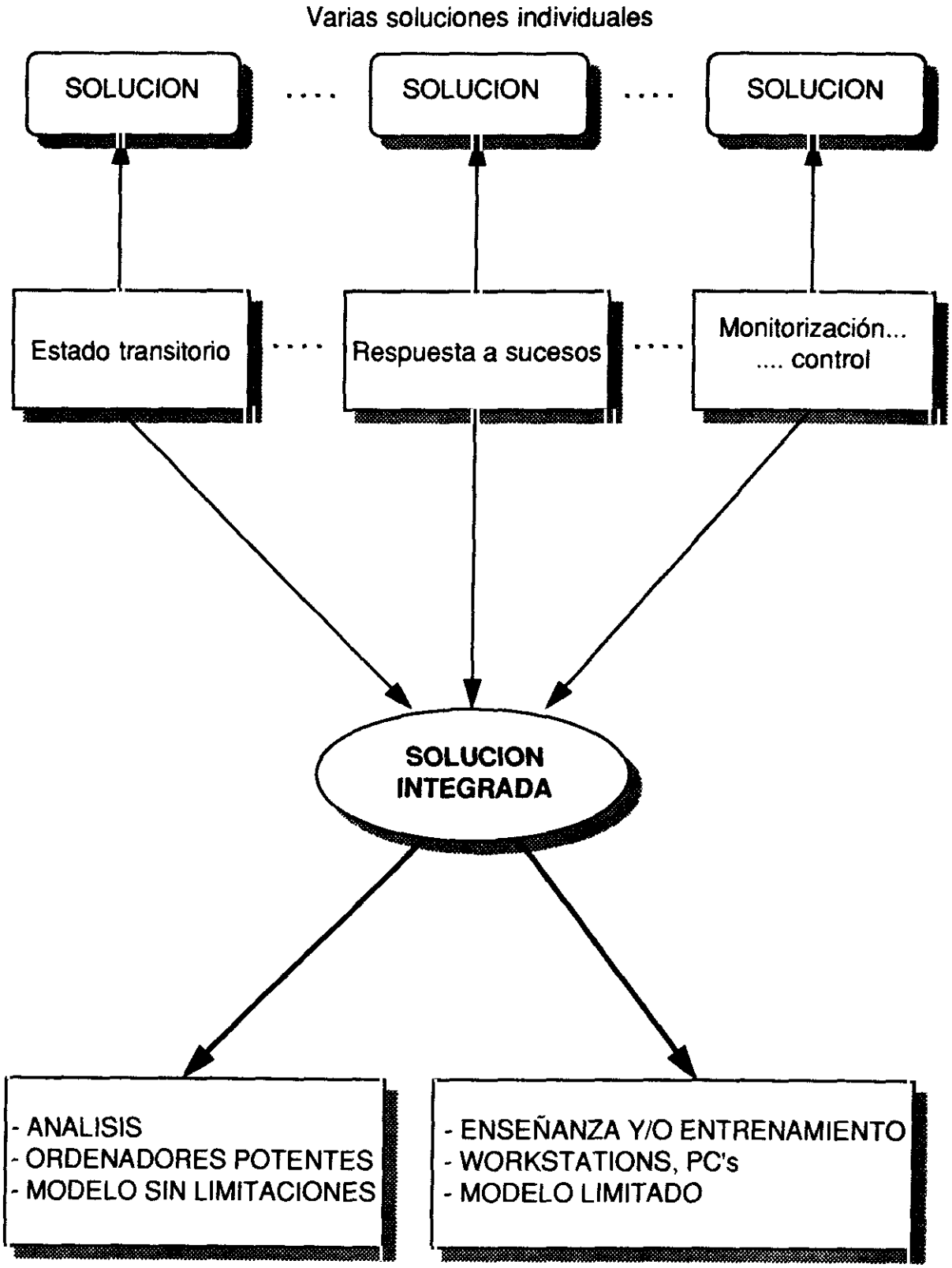

Varias soluciones individuales
SOLUCION
SOLUCION
SOLUCION
Estado transitorio
Respuesta a sucesos
Monitorización...
.... control
SOLUCION
INTEGRADA
- ANALISIS
- ORDENADORES POTENTES
- MODELO SIN LIMITACIONES
- ENSEÑANZA Y/O ENTRENAMIENTO
- WORKSTATIONS, PC's
- MODELO LIMITADO

Figura 1.3

Teniendo a la vista la dirección que hemos decidido, vamos a concretar el planteamiento de nuestra investigación, apoyándonos en los puntos que hemos discutido al revisar la bibliografía.

a) Planteamiento del Modelo.-

- **Modelación teórica.**
  Nos situamos en un nivel de abstracción previo a la toma de datos en una central que nos fuera accesible. Desarrollaremos un modelo propio, basado en el estudio de los fenómenos físicos que se producen. Nuestra atención se concentrará en el ciclo termodinámico que describe el fluido de trabajo, el agua, y no consideraremos los aspectos técnicos de, por ejemplo, si nuestra turbina es de tal o cual tipo, si tiene tantos o cuantos álabes, etc. Existirán limitaciones y simplificaciones en este modelo, al cual podemos de catalogar de sencillo, pero completo.

- **Modelación en variables de estado.**
  La principal problemática del estudio de las plantas de vapor y su modelación es la gran cantidad de parámetros en juego o variables, básicas para la descripción del comportamiento del sistema, y necesarias para la obtención de resultados. Teniendo en cuenta que el interés va menos al control y más a la descripción, y que el estudio está basado en formulaciones físicas y en tablas (fenómenos no lineales), que el sistema es complejo y la cantidad de variables elevada, entonces todo se decanta hacia las variables de estado, lo cual supone una filosofía concreta de tratamiento numérico.

  Por citar un ejemplo, diremos que el agua dentro de la caldera tiene unas características termodinámicas (presión, temperatura, densidad, entropía, etc) cuya medida y conocimiento son del todo imprescindibles para el cálculo de variables como el nivel en la caldera, gasto energético, etc, que son en última instancia las deseadas; Así, para hallar una variable han de calcularse muchas otras. El estudio mediante ecuaciones temporales se adapta mejor al cálculo numérico con muchas variables, con respecto a la alternativa que podríamos considerar, a base de funciones de transferencia.

- **Modelación de elementos individuales, nexados posteriormente en la planta.**
  Nosotros vamos a hacer los modelos de los dispositivos del prototipo de planta, preparándolos de manera que la salida de uno sea congruente con la entrada de otro; de esta forma, y en un nivel último, realizaremos el modelo de la planta

simplemente uniendo los dispositivos mediante conductos o tuberías ideales en los que las variables no pierden sus valores. Este tipo de modelación individual se corresponde con la utilización de una poderosa herramienta: la Programación Orientada a Objeto (POO). La especial adecuación de la POO al caso de la modelación y simulación, se explica porque las "clases" en la POO se asignan de una forma natural a los tipos de dispositivos, y los "objetos" a los elementos concretos; los "mensajes" son vehículos que soportan perfectamente las relaciones entre las distintas variables o sistemas ("clases", "objetos" y "mensajes" son términos característicos de la POO). Una ventaja de la modelación de los elementos discretos es que nos permitirá la posibilidad de elaborar modelos de conFiguraciones distintas de plantas (o sea, plantas distintas), multiplicando así la potencialidad de nuestras simulaciones. De seguir un tratamiento clásico, con un modelo único de la planta en su conjunto (es decir, un enorme sistema de ecuaciones) , ello no sería posible, siendo necesario elaborar otro modelo distinto para cada planta distinta.

- **Modelación dinámica.**
  Para simular el comportamiento a través del tiempo de un sistema, el modelo matemático debe reflejar la dependencia respecto al tiempo, a través de ecuaciones diferenciales y valores iniciales para una serie de parámetros. Así, podremos estudiar cómo serán los transitorios, cómo se alcanza el estado estacionario, y qué control es necesario efectuar para alcanzar dicho estado alrededor de unos valores de referencia.

b) Planteamiento de la simulación.-

- **Simulación basada en modelo propio.**
  Como hemos señalado, existen librerías profesionales para la simulación de sistemas térmicos, como el "Modular Modelling System", MMS, que son la base de numerosas simulaciones recogidas en la bibliografía ([Khadem, 1986], [Barcelo, 1985], [Smith, 1985], [Rosard, 1985]). Dados los objetivos y límites que nos hemos marcado (X-Windows, POO, etc.), no nos hemos planteado su uso. Por lo demás, su coste es prohibitivo, de no recibir una financiación de cierta envergadura. Nuestra decisión es elaborar un modelo propio, que puede constituír una posible fuente de originalidad.

- **Simulación interactiva.**
  Cabe precisar que algunos autores ([Kanodia, 1988], [Khadem, 1986]) consideran interactividad el poder elegir eventos predefinidos para que éstos se produzcan en el instante inicial de la simulación. Nosotros entendemos que el concepto de interactividad exige que los eventos se produzcan en tiempo de

simulación, y además en el instante que desee el operario. Otra cosa es que se consideren qué tipo de eventos pueden darse: cualesquiera o solo algunos definidos. En nuestro caso, y aprovechando la formulación en variables de estado de nuestro modelo, elegiremos una serie de variables que llamaremos de control, las cuales podrán modificar sus valores en tiempo de simulación (por ejemplo, la cantidad de calor proporcionado por un quemador, o el caudal suministrado por una válvula o bomba, o la constante de expulsión de una turbina, o la temperatura de entrada del agua refrigeradora en el condensador, etc). La variación de estas variables de control simula incidencias, incluso averías; por ejemplo, haciendo que el caudal de salida de una válvula vaya progresivamente disminuyendo, se puede representar la rotura de una tubería con pérdida de fluído. Otras variables (entre las que se cuentan las de control) podrán cambiar sus valores iniciales antes de comenzar la simulación. Un último grupo de variables de estado ajustarán sus valores, una vez comenzada la simulación, a los procesos físicos que se produzcan. Cabe añadir que esta interactividad se refleja en todo instante en las gráficas, y que se incluye también la posibilidad de parar una simulación para su posterior continuación, o bien una simple cancelación. Conjuntamente a la interactividad, se dota a la simulación con la capacidad de guardar simulaciones en ficheros para su posterior recuperación y visualización.

- **Simulación con control.**
  Nuestra simulación no cubre, por el momento, posibles lazos de control insertos en la planta. Ahora bien, mediante controles automáticos implementados en nuestro programa de simulación, daremos respuesta a una serie de problemas, evitando que algunas variables se disparen debido a la marcha del proceso. Por ejemplo, si la temperatura del vapor supercalentado alcanzase un valor que hiciera peligrar la fiabilidad de los cálculos, un control cerraría inmediatamente las espitas de los quemadores que calientan el supercalentador, de manera que la variable queda controlada y confinada a unos rangos de seguridad.

- **Simulación dedicada a la enseñanza y/o entrenamiento.**
  Nos proponemos la simulación de un prototipo de planta de vapor con características que la hagan útil y conveniente, entre otros usos, para aplicaciones didácticas. Este objetivo tiene importantes implicaciones en los niveles a alcanzar, las técnicas y herramientas a utilizar, y todo lo que dice a interacción amigable y funcionalidad. Un caso que nos imaginamos, es el de un futuro operario de plantas térmicas o sistemas afines, en una etapa introductoria y conceptual respecto al proceso y sus fundamentos, la monitorización de parámetros, la posibilidad de averías, etc.

I.2.2.- Enfoque y Programación del Trabajo.-

Una vez planteada la modelación y simulación del prototipo de planta de vapor, hemos de precisar etapas y características de nuestro trabajo.

a) Enfoque.-

Resumimos en la Figura 1.4 el enfoque de nuestro trabajo. Dividimos la tarea en dos grandes etapas.

La etapa inicial se centra en el estudio de ciclos termodinámicos. No en su generalidad, pues son muy conocidos, sino escogiendo el ciclo de Rankine (por su importancia y aplicabilidad en las centrales térmicas). Consideramos su comportamiento una vez establecido un régimen estacionario. Un objetivo importante de esta etapa es precisar, por vía de investigación y bajo la perspectiva de lo que nos interesa, conocimientos termodinámicos, y tratar de comprender la problemática del ciclo de vapor, sus posibilidades y limitaciones.

En la segunda etapa, más vasta y ambiciosa, llevamos a efecto el objetivo que nos hemos marcado, de modelar y simular un prototipo de planta de vapor incluyendo sus aspectos dinámicos.

b) Programación del Trabajo.-

En la Figura 1.5 podemos ver esquematizada la definición de objetivos para las dos etapas propuestas, de una forma muy resumida, y que a continuación explicaremos, también brevemente.

Para la etapa inicial, nos proponemos desarrollar un programa (RANKINE.EXE) ejecutable bajo el sistema operativo MS-DOS, y creado bajo el lenguaje C. Este programa ha de satisfacer los siguientes requisitos:

— Ser de una naturaleza didáctica, mostrando los estados y variables termodinámicas más importantes, procesos, transferencias energéticas, rendimientos, gráficas, etc.

— Que sea interactivo para los valores iniciales, de forma que el usuario pueda ver el mismo ciclo bajo unas condiciones distintas.

— Que cuente con ayudas y descripciones.

Para la segunda etapa, que es el centro de nuestro trabajo, nuestra aplicación a desarrollar deberá contar con los siguientes requisitos:

— Ha de ser una plataforma única.

— Ha de verse cómo evolucionan a través del tiempo las distintas magnitudes de interés.

— Han de poder ser manipuladas las variables de estado controlables en cualquier momento de la simulación, de manera que podamos percibir el efecto que produzcan estas variaciones.

— Ha de contar con ayudas y explicaciones.

— Ha de disponer de una manera de recuperar simulaciones previamente almacenadas en ficheros.

— Ha de tener un aspecto didáctico, de fácil uso y adecuada comprensión.

Como objetivo práctico nos propondremos la creación de sendos programas, PLANTA.EXE y PLANTA, para ser ejecutados bajo los sistemas operativos MS-DOS y UNIX respectivamente. Ambos programas, bajo C++: el primero, con TURBO-C++ de BORLAND, para plataformas PC, y el segundo, bajo GNU g++ y con entorno gráfico X-WINDOWS v11 R4, para estaciones de trabajo (nosotros hemos utilizado una SUN-SPARC). Nos proponemos hacer el mismo trabajo en estos dos programas debido a la repercusión deseada para la aplicación. Así, y debido a la amplia difusión de ordenadores PC bajo sistema operativo MS-DOS, los estudiantes podrán hacer uso de esta simulación, mientras que para ambientes más especializados, con "workstations" bajo X-Windows, se podrá contar con nuestra aplicación.

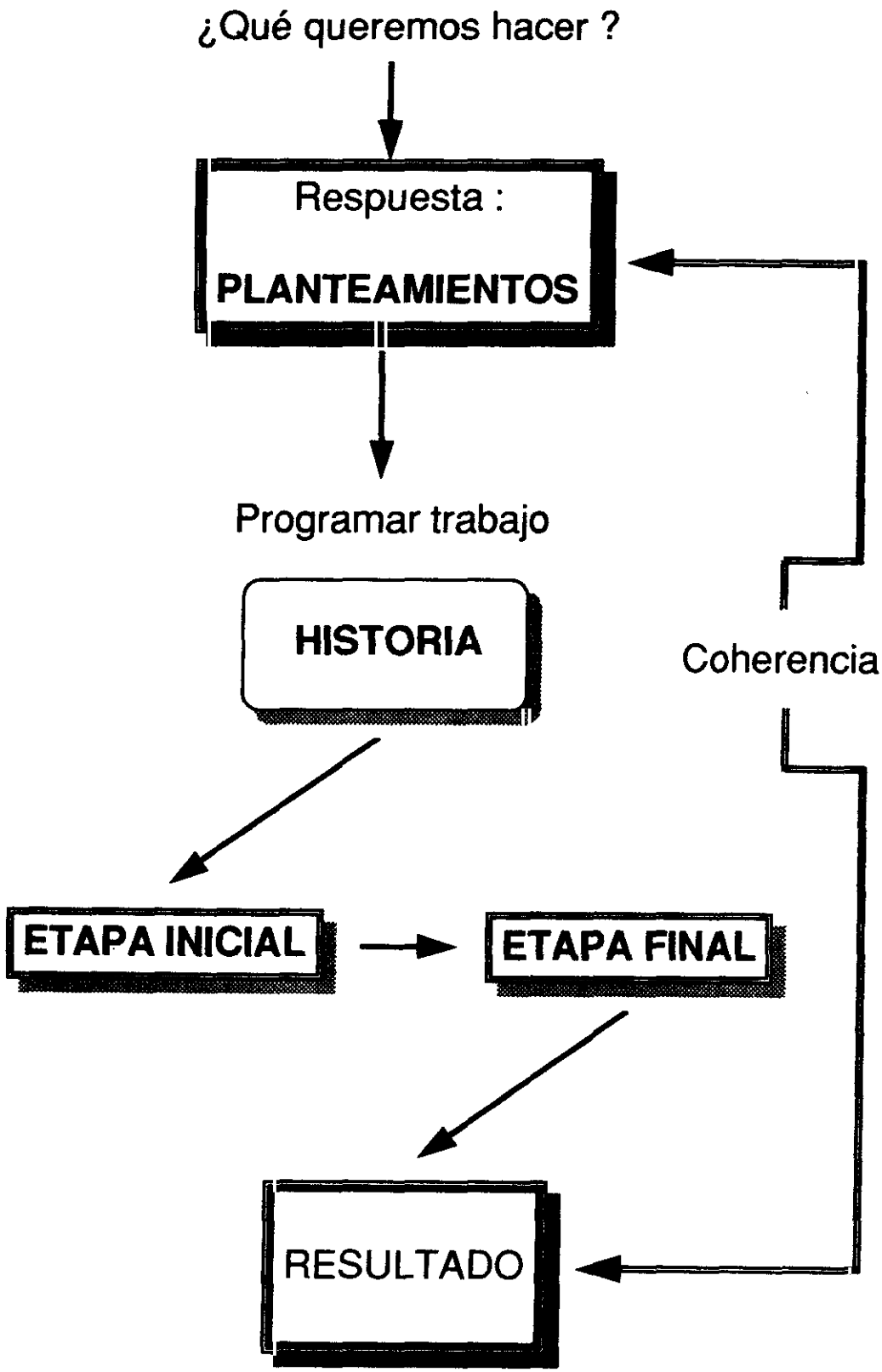

¿Qué queremos hacer ?
Respuesta :
PLANTEAMIENTOS
Programar trabajo
HISTORIA
Coherencia
ETAPA INICIAL
ETAPA FINAL
RESULTADO

Figura 1.4

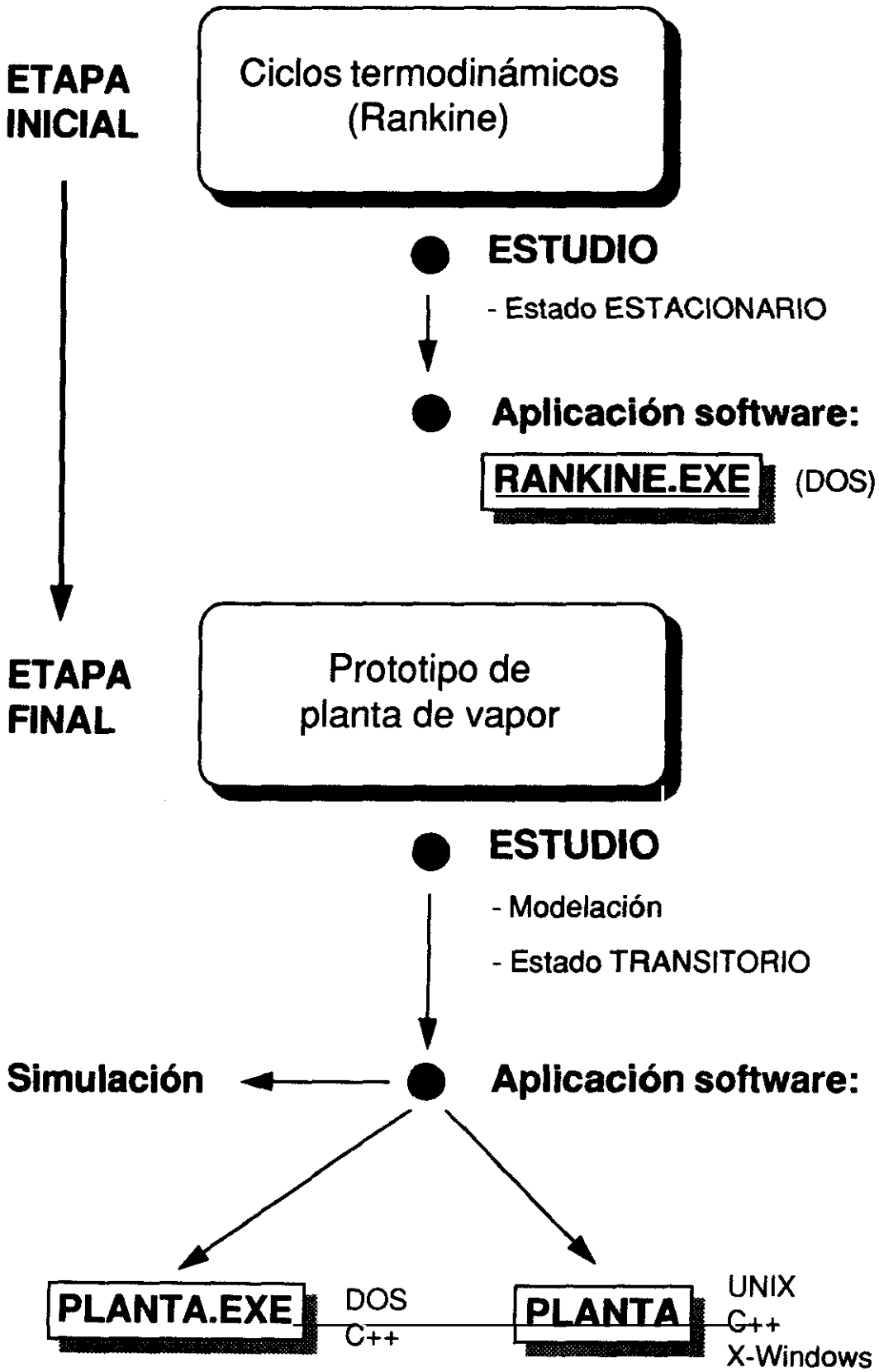

ETAPA INICIAL
Ciclos termodinámicos (Rankine)
ESTUDIO
- Estado ESTACIONARIO
Aplicación software:
RANKINE.EXE
(DOS)
ETAPA FINAL
Prototipo de planta de vapor
ESTUDIO
- Modelación
- Estado TRANSITORIO
Simulación
Aplicación software:
PLANTA.EXE
DOS
C++
PLANTA
UNIX
C++
X-Windows

Figura 1.5

I.2.3.- Elementos Originales. Dificultades.-

Resumimos aquí los aspectos originales que nuestra investigación posiblemente aporte en el contexto actual. Junto a ello, comentamos también brevemente las principales dificultades que se han afrontado.

Hemos realizado una modelación según la mentalidad radical de la Programación Orientada a Objeto. La tradición de ingeniería está dominada por el FORTRAN, y son muy pocos los autores que utilizan otros lenguajes. De ellos, algunos llegan a emplear C. En una ocasión reciente, en un Congreso Internacional de Simulación, en Nueva Orleans, se comentó el empleo de C++, como un C mejorado, manteniendo la metodología tradicional, sin cambiar a la mentalidad de la POO. Esta es una actitud conservadora típica, que se resiste a la aventura. La literatura especializada insiste en que, efectivamente, pasar a la Programación Orientada a Objeto supone un cambio sustancial de actitud y planteamientos (por eso, empleamos la expresión: mentalidad radical).

En este sentido, pensamos que una aportación de nuestro trabajo es mostrar las posibles ventajas de la POO en el ámbito de los simuladores de centrales, en cuanto a que ayuda a una modelación natural, y simplifica la modularidad e inteligibilidad del código desarrollado.

En cuanto a tecnología, deseamos contribuír al empleo de estaciones de trabajo para aplicaciones de simulación especializada, adhiriéndonos a una normalización (X-Windows) que potencia la capacidad gráfica y la fácil interacción de usuario.

A lo largo de la bibliografía consultada, no hemos encontrado ninguna referencia de modelación de centrales basada en POO. Por otro lado, son muy escasas las simulaciones de centrales que hacen uso de X-Windows (cada uno de estos casos requiere un comentario especial, porque las firmas comerciales intentan poner su propio marchamo, llegando a desarrollar por su cuenta clónicos de X-Windows). Sí hemos encontrado una referencia, [Berenbach, 1991], que trata la POO al problema de un simulador de planta, pero de manera colateral, es decir, sin aplicar la POO a la modelación, sino a cómo el simulador puede comunicarse en un entorno de red.

La mayor dificultad vino de la elaboración del modelo. La bibliografía es muy reacia a explicar los modelos que utilizan; más bien dicen poseer uno, extendiéndose sobremanera en los aspectos de la simulación. Además, de los escasos trabajos que explicitan sus modelos, prácticamente ninguno suele ser el adecuado, pues nos hemos planteado una serie de requisitos (variables de estado, dinámica, elementos individuales, etc) que nos limitan a la hora de aprovechar modelos publicados. Por tanto, podemos concluir que el modelo es en un 50% propio (condensador, válvulas, bombas,

tuberías y quemadores) y un 50% ajeno (caldera, turbina y supercalentador), del cual tuvieron que modificarse y replantearse aspectos para una correcta integración. Esta contribución ajena descansa, principalmente, en [Usoro, 1977] y [Masada, 1979]. En el capítulo dedicado a la modelación se desarrollarán ampliamente las explicaciones de los modelos, comentando las dificultades, limitaciones, supuestos, etc.

Además se han debido superar las lógicas dificultades para asumir la nueva mentalidad propuesta por la Programación Orientada a Objeto; y para el dominio de UNIX y X-Windows, lo que requiere un serio esfuerzo de aprendizaje y uso de manuales.

Citemos también ciertas dificultades para conseguir e instalar software especializado (en especial, GNU g++ y X-Windows v11 R4) sobre plataformas UNIX (Sun-Sparc y PC-80386 con sistema ESIX), entre otras razones, debido a la escasa documentación.

I.3.- REVISION BIBLIOGRAFICA.-

Pretendemos en este tema dar una visión panorámica de la bibliografía a la que hemos tenido acceso. Podemos, a efectos descriptivos, dividirla en tres partes:

I.3.1.- Bibliografía de Propósito General.-

Este primer bloque bibliográfico, el más extenso y numeroso, ha cumplido un importante papel orientativo, para conocer el estado de la cuestión (qué aspectos se han tratado, con qué técnicas se han contado, cuáles son los intereses, etc.), y hacer un planteamiento de nuestro trabajo de acuerdo con las posibles carencias encontradas. Especialmente nos han interesado varios artículos y ponencias de Congresos Internacionales, tres libros y dos tesis.

Una precisión a hacer es que, pese a no interesarnos las centrales nucleares, sino las plantas de vapor, hemos considerado algunos trabajos de ese ámbito por sus características de modelación y simulación, bastante próximas a lo que nos interesa. A continuación, daremos una rápida visión de estas fuentes bibliográficas de propósito general:

- [Kanodia, 1988], desarrolla un programa para el entrenamiento en plantas. Posee una base de datos, un editor gráfico y ejecuta después de dar valores iniciales. Podemos destacar su editor gráfico de planta, con el cual el usuario puede hacer distintos diagramas de flujo para su simulación. No aporta información alguna sobre el modelo, y presenta una pobre presentación gráfica. El nivel de interactividad durante la simulación permite la introducción de eventos predefinidos en el programa.

- [Khadem, 1986] concibe un simulador para entrenamiento de operarios. Podemos catalogarlo de *full-scope* por sus características, que incluyen buenos gráficos, dinámica completa e interactividad, la cual se basa en la selección inicial de eventos para la simulación de averías. Trae dos ejemplos, una planta nuclear y otra térmica de combustión. El modelo descansa en librerías MMS,

simulado bajo FORTRAN 77 sobre una mini DEC-VAX con dos consolas: una, en blanco y negro para la introducción de selecciones a base de menús (instructor), y otra en color para la monitorización gráfica.

- [Stegemann, 1986] es de los pocos que describen el modelo, mediante ecuaciones analíticas. Es un modelo de elementos individuales (reactor, generador de vapor, presurizador) preparado para poder simular una planta entera, conectándolos por tuberías. El modelo, que se basa en plantas reales (aunque teórica, no experimentalmente), se simplificó deliberadamente para ofrecer las ecuaciones analíticas. La simulación es de los elementos separados, no de la planta entera; simula dinámicamente, con perturbaciones. Fué desarrollado en FORTRAN IV sobre una máquina CYBER 76, dando una relación de tiempo de simulación por tiempo de computación de siete.

- [Saphier, 1986] se limita a ofrecer un lenguaje orientado-a-bloques, el DSNP, para simular sistemas; tiene modelos de componentes en la librería, de manera que al construir un diagrama, lo simula dinámicamente después de dar unas condiciones iniciales.

- [Li-Chi, 1986] considera un solo elemento (de una planta nuclear), el reactor. El modelo es completo, sofisticado, muy detallado físicamente a través de sus ecuaciones diferenciales, y se divide en sub-elementos. La simulación es completa, concebida para el entrenamiento, con gráficos en color, control interactivo y simulando transitorios o accidentes definidos.

- [Babula, 1986] hace un planteamiento análogo al de [Li-Chi, 1986], pero sin entrar en la simulación. La implementación del modelo la hace en el lenguaje RIGEL.

- [Belblidia, 1986] sigue en la línea de los dos anteriores, si bien el elemento (un generador de vapor, en un artículo, y un presurizador, en otro) queda preparado para integrarse, junto con otros elementos, en una planta, a través del lenguaje DSNP de [Saphier, 1986].

- [Kwatny, 1986] modela solamente los quemadores, en un trabajo teórico bien explicado, a través de ecuaciones diferenciales. No considera el control.

- [Skanit, 1986] aporta el control a un generador de vapor en un modelo teórico, dinámico, para alcanzar un *set-point* del nivel que se desea en el estado estacionario.

- [Chaplin, 1986] también considera el control en su modelo, el cual está basado en una planta real; así, ajusta la teoría a los datos obtenidos de la planta. La simulación se basa en los lenguajes BASIC y FORTRAN sobre máquinas Apple IIe y Zenith Z-150.

- [Zwingelstein, 1986] no investiga modelo alguno, sino que su aportación consiste en el desarrollo de un sistema numérico interactivo para la simulación, y que, tras tomar datos de la planta, identifica para una posterior simulación, optimizando, describiendo señales, etc.

- [Bernard, 1986] toma datos del reactor creado en el M.I.T., con los que analiza, controla y simula dicho sistema.

- [Rajakumar, 1986] incide nuevamente en cómo un modelo matemático ha de simplificarse para adecuarlo a la realidad de un reactor en el cual se basa. Es un estudio completo que incluye control y simulación de incidencias.

- Citamos a [Barcelo, 1985], [Smith, 1985] y [Rosard, 1985] conjuntamente para explicar cómo la librería MMS (Modular Modeling System) permite hacer numeros desarrollos sin tener que modelar cada elemento. Sobre MMS hablaremos en el próximo capítulo.

- [Ocampo, 1984] modela teóricamente una turbina, ajustando el modelo posteriormente a una turbina real mediante la toma de datos. Prepara el modelo para integrarlo luego en una planta, después de simularlo y testearlo.

- [Kubiak, 1984] también modela turbinas, pero en el dominio "s" para el control. El procedimiento es partir de una descripción física, simplificar adecuadamente y después linealizar, obteniendo así el modelo en funciones de transferencia.

- [Dunne, 1984] es un ejemplo de modelación del ciclo de vapor en el estado estacionario. Modela en estado-estacionario cada elemento, integrando los bloques posteriormente en una planta térmica.

- [Roldán-Villasana, 1984] modela una caldera teóricamente, simplificando para adecuarlo a un sistema real.

- [Riley, 1984] ofrece la hasta ahora novedad de modelar en un solo bloque una planta entera, y además, en estado estacionario. Mediante un sistema llamado KRAB, el usuario puede generar un modelo de planta entera en estado estacionario (TPM, Steady-State Total Plant Model) para unas especificaciones, sin necesidad de escribir ningún programa.

- [Berenbach, 1991] es el único de los que hemos podido leer que acerca un simulador de planta térmica al ámbito de lo orientado-a-objeto; no es el modelo lo que está en POO, sino que el simulador, ya desarrollado antes, se mete en un entorno POO para comunicación con estaciones, etc. Esto nos confirma cierta originalidad al desarrollar nuestro trabajo un modelo y una simulación orientada a objeto.

- En un libro, [Knowles, 1990] explica varios modelos dinámicos en variables de estado, que implementa mediante técnicas matemáticas. No hemos tomado apoyo ninguno en este libro pues su modelo no nos era atractivo para una implementación en POO; sin embargo, lo ofrecemos al lector pues es raro en los artículos y ponencias extenderse en la explicación de los modelos.

- [Polony, 1991] ,también en otro libro, modela en funciones de transferencia elementos de la planta. Así, y en el dominio "s", hace unos estudios sobre el control de la presión en la caldera, y el control del nivel en la misma.

- [Weisman, 1985] ofrece a través de un libro una visión panorámica de la problemática general de las plantas térmicas; es un libro más de referencia didáctica que de desarrollos específicos. Sin embargo, lo encuadramos en esta sección pues implementa modelos matemáticos para su simulación.

- [Masada, 1979], a través de una extensa Tesis Doctoral desarrollada en el M.I.T. modela matemáticamente, basándose en las leyes físicas, y con técnicas de parámetros distribuidos, elementos de una planta real con la cual puede establecer comparaciones. La simulación de incidentes se especifica al principio de la misma.

- Por último, en la Tesis Doctoral [Azuma, 1975] podemos encontrar una referencia análoga a la de otros autores antes mencionados que consideran el aspecto del control de una planta, mediante un desarrollo en funciones de transferencia. Esta tesis ofrece la totalidad de la planta.

I.3.2.- Bibliografía Especializada.-

El presente grupo bibliográfico reside especialmente en libros, en los cuales hemos buscado, por una parte, referencias al estudio termodinámico de la planta, y por otro, conocimientos de las herramientas necesarias para nuestro desarrollo.

- Citamos conjuntamente a [Howell, 1987] y [Joel, 1987] como dos libros de ingeniería termodinámica, dirigidos a estudiantes de algún curso superior de esta especialidad. Describen desde los fundamentos básicos de la termodinámica, hasta una descripción detallada de los ciclos y máquinas térmicas. El primero ofrece un interesantísimo apéndice de **tablas de vapor** que han sido fundamentales para nuestro trabajo, así como el capítulo dedicado a estudiar el ciclo de Rankine.

- [Reynolds, 1979] y [Grigull, 1990] son dos libros que ofrecen las tablas de vapor en el sistema internacional de unidades (S.I.). Hubo que integrar las tablas, tanto de estos dos autores, como las de [Howell, 1987] en nuestro trabajo pues, en algunos estados como el de vapor supercalentado, las de Howell llegaban a unos límites que eran superados por las de Reynolds. En síntesis, las tablas de los estados líquido saturado, vapor saturado, y liquido provienen de Howell, y las de vapor y vapor supercalentado, de Reynolds.

- Los manuales de programación adjuntos a los paquetes "Turbo-C v2.0" y "Turbo-C++ v1.0" de Borland International Inc. fueron usados para el aprendizaje tanto de los lenguajes C y C++ respectivamente, como de los correspondientes entornos de programación (que incluyen compiladores, editores, depuradores, etc.).

- Sin embargo, [Zimmerman, 1989] es una guía completa y didáctica del lenguaje Turbo C y de su correspondiente entorno, más adecuada para el principiante, que pronto buscará en los manuales de Borland todo cuanto necesite para su programación.

- Igualmente diremos de [Voss, 1990], que ofrece al programador en C de la manera más comprensible de pasar a programar en C++, es decir, orientado a objeto. Tras este conocimiento, los manuales de Borland serán el siguiente paso.

- Tanto para el Turbo C como para Turbo C++, el libro [Weiskamp, 1989], ofrece una extensa visión sobre las librerías C de Borland acerca de gráficos, manejo de textos, sistemas Input/Output, etc., más difícil de encontrar bien explicadas y con ejemplos en los manuales de Borland.

- [Barkakati, 1991] y [Johnson, 1989] son dos libros para estudiar los entornos de programación en X-WINDOWS. El primero ofrece un trabajo muy completo, desde la programación de todos los aspectos como creación de ventanas, gráficos, colores, textos, eventos, etc. hasta la utilización de los X-Toolkits, OSF/MotifWidgets, aplicaciones avanzadas, etc. El segundo es más apropiado para el principiante, limitando muchos aspectos del conocimiento de X-WINDOWS. Podemos decir así, que Barkakati es una completa guía para un programador ya experimentado en este entorno, y Johnson es ideal para el que comienza.

I.3.3.- Bibliografía de Ayuda.-

Destacamos una tesis y un extenso artículo, que nos han sido especialmente importantes en cuanto a la obtención de ayudas para nuestro modelo.

- [Usoro, 1977] desarrolla en su tesis doctoral unos modelos teóricos en variables de estado, que se basan en dos técnicas: un modelo, llamado "standard", se basa en las puras leyes físicas, y otro, llamado "digital", en la técnica de parámetros distribuidos, teniendo pues éste último menos variables y un tamaño, en general, más reducido. Nos ha servido de ayuda esta tesis para desarrollar los modelos del supercalentador y de la turbina, los cuales hubo de adaptarlos, tanto para su programación orientada a objeto, como para su integración con otros elementos constitutivos de la planta. A través de esta ayuda, tenemos una cierta **referencia experimental** de nuestro modelo, pues Usoro se basa en sistemas reales existentes en el M.I.T.

- [Dieck-Assad, 1990] nos ha sido especialmente útil a la hora de desarrollar un modelo de caldera en variables de estado. Explica bien el sistema de producción de vapor, resumiendo posteriormente otros sistemas. La simulación que realiza es de la planta entera.

Capítulo II:

BASES TECNOLOGICAS

Capítulo II:

BASES TECNOLOGICAS

En este Capítulo queremos dar una visión general sobre las herramientas de simulación (lenguajes, paquetes informáticos, etc.), para después ver en detalle las bases tecnológicas de nuestro trabajo.

Dividimos la exposición en tres apartados.

En el primer apartado, repasaremos las tecnologías tradicionales en la simulación de las plantas térmicas, según dos aspectos: las librerías específicas para estas plantas, y por otro, las herramientas generales para la simulación y control de procesos, que cabe utilizar en el tema que nos ocupa.

Estudiaremos en el segundo apartado metodologías de reciente aparición, tales como la Programación Orientada a Objeto y los entornos de ventanas y ratón.

Concretaremos en el tercer apartado la base tecnológica de nuestro trabajo, describiendo los equipos y software utilizados, sus características, utilidad, aplicabilidad, etc.

Con este Capítulo cerramos el espacio que hemos dedicado a revisar el estado de la cuestión en los diversos aspectos que nos atañen.

II.1.- TECNOLOGIAS TRADICIONALES EN LA SIMULACION DE PLANTAS TERMICAS.-

Por tradicional no queremos decir, ni muchísimo menos, anticuado. Tradicional tiene sentido para nosotros en cuanto se habla de técnicas que tienen una amplia implantación desde hace años y que han generado multitud de aplicaciones. Estas técnicas son más o menos recientes; algunas, con sus años a cuestas, van cayendo en desuso, mientras que otras se encuentran en el cénit de su implantación. En cualquier caso, son las que han sustentado y sustentan en la actualidad los trabajos de simulación.

Vamos a efectuar una revisión comentada del software disponible para simulación de centrales. Consideraremos dos secciones. En la primera, daremos un repaso a los sistemas de librerías clásicas para la simulación de plantas térmicas. En el segundo, veremos herramientas generales (paquetes, entornos, lenguajes) usadas en la simulación de sistemas, que pueden ser aprovechadas por el investigador para desarrollar sus aplicaciones.

II.1.1.- LIBRERIAS PARA LA SIMULACION DE PLANTAS TERMICAS.-

Básicamente, hablaremos aquí de MMS y de EASY5x; la primera es la librería de módulos de plantas y sistemas termodinámicos por excelencia, generadora de multitud de investigaciones, mientras que la segunda, con una librería en este campo más reducida, introduce la peculiaridad de dar técnicas generales para la simulación.

- El **Modular Modeling System** (**MMS**), de la firma norteamericana **Babcock & Wilcox** es una librería de una importancia ya histórica en el ámbito de la simulación de plantas térmicas. Mediante ella, los investigadores pueden ejecutar una gran variedad de análisis de transitorios y sistemas de control para sus plantas. Está basada en ACSL.

En 1978, el Electric Power Research Institute (EPRI) sacó un concurso para la creación del MMS, que correspondió principalmente a Babcock & Wilcox. Se desarrollaron más de 100 módulos representando la mayoría de los componentes usados en las plantas térmicas y nucleares. Todo ello, con un adecuado soporte de entrenamiento en teoría, fundamentos de código, desarrollos de modelos de plantas específicas, interpretación de resultados, etc. Pronto alcanzó un alto grado de comercialización, lo cual posibilitó el desarrollo de aplicaciones en todo el mundo. Esto se organizó de manera que existiera un Grupo de Usuarios, encargado de coordinar la asistencia técnica, enseñanza, etc. En España, la organización Empresarios Agrupados, de Madrid, usuario de MMS, llevó a cabo estudios de evaluación de funcionamientos de plantas, optimización de sistemas de control, aplicaciones de simulación para plantas térmicas y nucleares, etc.

- **EASY5x** (Engineering Analysis Software) de la norteamericana **Boeing Company**, es más un sistema integrado de herramientas para la modelación y simulación en general, que una pura librería de componentes para plantas térmicas como es MMS. Sin embargo, lo consideramos en este apartado y no en el siguiente pues contiene librerías específicas entre las cuales hay una de elementos de estas plantas. EASY5x es una versión de EASY5 para X-Windows, lo cual le otorga un fuerte impacto interactivo y visual sobre las estaciones de trabajo. Así, desarrolla un interface gráfico de usuario a base de ventanas, menús, botones, barras, ratón, etc. Puede modelar, simular, analizar y diseñar virtualmente cualquier tipo de sistema dinámico, contínuo o digital, caracterizado por ecuaciones diferenciales algebraicas, lineales, etc.

Las herramientas de diseño y análisis que ofrece incluyen simulación no-lineal, análisis en estado estacionario, generación de modelos lineales, respuesta en frecuencia, lugar de las raíces, margen de estabilidad, simulación lineal, optimización de parámetros, etc. En cuanto a sus gráficos de salida, tiene respuesta en el tiempo, trazos de Bode, carta de Nichols, diagramas de Nyquist, tablas de datos, etc. Las librerías se dividen en librerías de propósito general, control del entorno, hidráulica, ciclo de vapor, aerodinámica, etc.

II.1.2.- HERRAMIENTAS GENERALES DE SIMULACION.-

Dado que la simulación es un sector en expansión, impulsado por el buen precio y potencialidad de los ordenadores, no es extraño que aparezcan con relativa frecuencia nuevas herramientas y aplicaciones, junto con mejoras de lo ya existente. El catálogo publicado por la Society for Computer Simulation (SCS) registra más de 200 programas de simulación, en diversas categorías. A quienes estamos dedicados a Control Automático, nos interesa especialmente el repertorio de productos, del orden de 20, que mantiene RAPID DATA, que incluye herramientas de simulación de indudable valor académico e industrial. No pretendemos aquí efectuar una revisión exhaustiva de toda la oferta actual de herramientas de simulación, que sería demasiado extensa y algo reiterativa; sino más bien comentar algunos ejemplos relevantes.

Conviene notar que frecuentemente los autodenominados "lenguajes" de simulación, se construyen sobre la base de otro (p.ej. FORTRAN) que le presta el mecanismo de traducción a lenguaje máquina.

- **ACSL** (Advanced Continuous Simulation Language), de MGA Inc. es un lenguaje de simulación para modelar la respuesta dinámica de sistemas físicos. Este lenguaje ha sido ampliamente difundido, creando un espectro de aplicaciones que incluye la aeronáutica, control, instrumentación, química, termodinámica, biología, etc. Está considerado como un standard dentro de los paquetes de modelación de sistemas dinámicos. Entre otras muchas cosas, ofrece una gran variedad de algoritmos de integración, gráficos de Bode, Nyquist, lugar de las raíces, determinación del estado estacionario, etc.

  El lenguaje está basado en FORTRAN, de manera que admite librerías FORTRAN generadas por el usuario.

  Mediante ACSL se han creado, a su vez, paquetes y librerías como MMS, SIMUSOLV, etc. Existen también interfases gráficas, como PROTOBLOCK, para trabajar con él.

  Citemos que existen interfases que conectan ACSL con MATLAB y otras herramientas, ampliando la potencialidad de este lenguaje para la optimización, diagramas de bloques, identificación de sistemas, análisis matricial, estimación, gráficos tridimensionales, etc.

Está diseñado para trabajar sobre un amplio abanico de estaciones de trabajo. Su precio era en 1988, orientativamente para una workstation SUN, de 8.000 $, llegando a alcanzar, para otras máquinas, hasta los 39.000 $.

- De origen académico, **ESL** es un lenguaje de simulación basado en FORTRAN 77, integra ecuaciones diferenciales, tiene interfase gráfico para generación de modelos y simulaciones. Puede correr en las principales estaciones de trabajo.

- La firma **CACI**, muy activa en el campo de la simulación de propósito industrial y de organizaciones, ofrece en la actualidad los siguientes paquetes:

  — **SIMSCRIPT II.5**, lenguaje de simulación, con buenas prestaciones gráficas, utilizado actualmente por más de 5.000 usuarios.

  — **SIMGRAPHICS**, es un paquete para la enseñanza, que ofrece gráficos animados del sistema que se simula.

  — **NETWORK II.5** , simulación gráfica animada de enlaces entre ordenadores.

  — **LANNET II.5**, simulación gráfica animada de redes locales.

  — **COMNET II.5**, simulación gráfica animada de redes de comunicaciones (incluído satélites).

  — **MODSIM II**, es un nuevo lenguaje de simulación orientado a objeto, con buenas prestaciones gráficas que incluyen animación.

  — **SIMFACTORY II.5**, simulación gráfica animada de plantas de manufactura.

- **CINEMA**, de **Systems Modeling Corp.**, es un sistema software/hardware para la simulación animada de un modelo; éste se crea con el lenguaje de simulación SIMAN, tras lo cual el sistema gráfico lo dota de animación sin necesidad de programar. Hay versiones disponibles tanto para PC's como para workstations SUN, VAX y Apollo.

- **G2**, de la empresa **Gensym**, es una herramienta para la creación de sistemas expertos en tiempo real, basados en "frames". Diseñado para grandes aplicaciones donde cientos o miles de variables son monitorizadas de forma concurrente, las aplicaciones típicas del G2 están en el control de procesos, fabricación flexible, etc. En el campo de la simulación, ofrece un interface llamado **GSPAN**, que es capaz de comunicar el sistema experto con simuladores externos, multiplicando así la potencialidad de las simulaciones. De un elevado precio, este sistema ha alcanzado amplia difusión.

- Es ampliamente conocida, en el mundo de la simulación, la intensa actividad de la firma **The MATH WORKS Inc.** en torno a MATLAB.

  El lenguaje **MATLAB**, de uso muy extendido en el ámbito académico y de aplicación, se ofrece en diversas versiones junto con un variado e interesante abanico de librerías. En la actualidad podemos disfutar de las siguientes posibilidades:

  — Control Systems Toolbox

  — Optimization Toolbox

  — Robust Control Toolbox

  — mu-Analysis and Synthesis Toolbox

  — System Identification Toolbox

  — Signal Processing Toolbox

  — MMLE3 State-Space Identification Tool

  — Neural Network Toolbox

  — Spline Toolbox

  — Chemometrics Toolbox

  Inicialmente bajo el nombre **SIMULAB**, ahora cambiado a **SIMULINK**, la citada firma ofrece un paquete para la modelación, simulación y análisis de sistemas dinámicos (que pueden ser no lineales), basado en un amigable entorno gráfico, mediante diagramas de bloques. Puede concebirse como una extensión de MATLAB, adaptado a las posibilidades de X-Windows, Macintosh y MS-Windows. Corre en las principales estaciones de trabajo.

- **X-MATH**, de **Integrated Systems Inc.**, obedece a principios y objetivos similares a los de MATLAB (matrices, orientación a control, etc.). Corre bajo X-Windows, con buenas prestaciones gráficas. Admite inclusión de código en C o en FORTRAN. Para completar el producto, en cuanto a la posibilidad de programar en modo gráfico los modelos, la citada firma ofrece MATRIXx/SystemBuild. El precio para estaciones de trabajo ronda los 10.000 $.

- **TUTSIM**, de la empresa holandesa **Meerman Automatisering**, es un software para la simulación de sistemas dinámicos sobre PC's. Es especialmente conocido por quienes emplean "bondgraphs" como herramienta de análisis para modelación.

- **SIMUL_TR**, de la empresa austriaca **Simutech**, es un sistema de simulación para sistemas dinámicos discretos y contínuos. El modelo se implementa tanto gráfica como textualmente. Ofrece herramientas de análisis, modelación y simulación discreta, etc. Se trata de un lenguaje orientado a compilador basado en C. Tiene un buen nivel gráfico, que incorpora también la animación. Opera sobre redes de "transputers", que son unos micro-procesadores de 32 bits sobre una arquitectura RISC. Alrededor de este producto, la citada firma ofrece las siguientes extensiones:

  — **PROSIMUL_R**, para logística y líneas de producción.

  — **SIMDRAW**, modelación jerárquica gráfica.

  — **BAPS**, modelación "bondgraph" no lineal.

  — **BAPSDRAW**, modelación "bondgraph" jerárquica gráfica.

- **CHINTZ**, de la compañía norteamericana **Chiltern Industrial Controls Ltd.**, es un paquete software que nos permite crear un interface a base de ventanas, símbolos, botones, alarmas, etc. para equipar nuestra aplicación de control (PLC's, PID's, etc.) o de simulación. Es pues, básicamente una herramienta para la visualización de nuestra aplicación. Funciona bajo IBM PC's y compatibles, necesitando mínimo de 640 Kb RAM, lo que le hace muy versátil y adecuado para todo tipo de necesidades.

- **ADSIM**, de la empresa británica **ADI** (Applied Dynamics International), es un lenguaje de simulación de propósito general, aparecido en 1989. Debido a su estructura, los modelos descritos por el usuario son simulados a alta velocidad. La notación que el usuario ha de manejar se basa en la notación matemática de alto nivel que normalmente se emplea en la ciencia. Con una extensa librería de funciones y modelos que facilitan el manejo de sistemas complejos, ADSIM ofrece un entorno gráfico atractivo, que incluye un completo control de la simulación.

- Otro lenguaje es **SYSL** (System Simulation Language), que permite la simulación de sistemas de todo tipo, mediante la solución de ecuaciones algebraicas y diferenciales que modelan los procesos dinámicos. Con una gran veteranía (desde 1975) ,SYSL trae su librería de funciones escrita en FORTRAN.

- **CYPROS**, de la noruega **CAMO** (Computer-Aided Modelling A/S), es una herramienta bastante útil, pues consiste en 18 programas interactivos que manejan el modelado matemático de sistemas, simulación, diseño de sistemas de control, adquisición de datos, análisis, procesamiento de señales, cálculos matriciales, etc. Concebido para PC y minis, evita el tener que programar para efectuar un buen número de tareas relacionadas con el ámbito del modelado, simulación y tratamiento de datos.

- **PROSIMULATOR** es un paquete software para la simulación y control de procesos en general. Básicamente, permite observar la dinámica de un proceso, creando un diseño para simularlo y, posteriormente, optimizarlo. Es de un sencillo manejo, y sus requisitos hardware (mínimo PC XT, 640Kb RAM, DOS 2.1, CGA) lo hacen versátil y aplicable especialmente para estudiantes.

- **ISI** (Intelligent Simulation Interface), de la firma irlandesa **Extech International**, es una herramienta de generación de código gráfico para PC. Básicamente, se trata de generar los códigos de unos lenguajes de simulación (SIMAN y SLAM-II) mediante un procedimiento gráfico que permite al usuario construir (ayudado por un interface de menús guiado por herramientas CAD, ratón, macros, ayudas on-line, etc.) sus modelos, de una manera rápida, sin necesidad de programar la simulación. Disponible para IBM PC/XT, AT, PS/2 y compatibles, los modelos pueden ser exportados a los conocidos paquetes de Auto-CAD y Lotus 1-2-3.

- **SIMUSOLV**, de la norteamericana **MGA Inc.**, es un software de modelado y simulación que utiliza ACSL para la especificación del modelo. La forma de construir el modelo es como sigue: se postula un modelo, escrito en ACSL; posteriormente se simula, estimando los mejores valores de los parámetros; entonces se discrimina si el modelo es el apropiado, para simularlo de nuevo, pero ya optimizado, o bien se busca un nuevo modelo si el anterior no es el adecuado.

- **CODAS** (Control System Design and Simulation) y **PCS** (Process Control Simulation), de **Golten & Verwer Partners**, son paquetes para el diseño y simulación de sistemas de control el primero, y para la simulación de procesos bajo tres tipos de control el segundo. Ambos, para PC.

- **SIMNON/PC**, de origen académico (una investigación centrada en la simulación de sistemas no-lineales), y representada ahora por **SSPA Systems**, es un entorno de simulación de propósito general, para predecir la dinámica de un proceso dados los supuestos sobre su comportamiento interno y la interconexión de sus componentes. Las ecuaciones que representan los componentes y conexiones pueden ser lineales o no-lineales, operando en tiempo contínuo o tiempo discreto. Es disponible para PC's.

- **PIM+**, de **Adaptech**, es un paquete interactivo para la identificación de sistemas a partir de unos datos experimentales, diseño y simulación de controladores, y elaboración de gráficos científicos. Los modelos identificados pueden ser transferidos a otros programas, como Program CC y MATLAB.

- **MODEL**, de **Model Software**, es un lenguaje orientado a objeto de alto nivel que permite la simulación de procesos distribuidos en tiempo real; incluye su propio compilador, preprocesador, entorno de utilidades, etc. Tiene una extensa librería de funciones, entre las que destacamos tablas de vapor. Se ejecuta bajo PC compatibles.

- **SIMBOL 2**, de la británica **Cambridge Control**, es un modesto programa interactivo de simulación y análisis para el diseño de sistemas de control, para IBM PC ó PS/2.

- **Program CC** es una asequible herramienta para el diseño de sistemas de control asistidos por ordenador, para IBM PC's y compatibles. Como herramienta de ayuda a la simulación, podemos introducir nuestro modelo, descrito linealmente o en espacio de estados, para su simulación y control. Está escrito y compilado en BASIC.

En el catálogo actual de RAPID DATA figuran, además de ACSL, MATLAB, SIMULINK, PROTOBLOK, MMS, SIMUSOLV, que ya hemos citado, otros paquetes que sólo mencionaremos: ACSL/GRAPHICS MODELLER, OPTDES, EASE+TOOLS, ACSL/MATLAB UTILITIES, CAMP, SD/FAST, MICROSAINT, DADiSP, MATHEMATICA, ORIGIN, SIGMAPLOT, PROCEDE, DART, ACET, ESIM, VISSIM, etc.

Conocemos además, por otras fuentes, la existencia de paquetes y lenguajes como: DESIRE, DYNAST, MAPLE, MATHCAD, ESACAP, EXTEND, FSIMUL, HYBSYS, STEM, XANALOG, CTRL-C/MODEL-C, AIM, KBVISION, SLAMII,. Por su relación con el lenguaje C, nos parecen de especial interés, PROSIGN (de Helmuth Stahl, Germering), SIMFLEX/2 (Universidad de Kassel). No se ha de olvidar el clásico GPSS (existen versiones, como el GPSS/H que admiten gráficos animados), con el que guardan relación GSS y diversas extensiones.

Recientemente ha aparecido SIMPLE++, desarrollado en Stuttgart por el Instituto de Investigación IPA-FhG, escrito en C++, corre bajo X-Windows, bastante orientado a cuestiones de logística y manufactura.

La oferta actual de herramientas de software está creciendo especialmente en sectores relacionados con eventos discretos (manufactura, redes y comunicaciones, logística, etc.), y temas de importancia estratégica, de infraestructura y financiación.

II.2.-TECNOLOGIAS ACTUALES.-

Aparte de los lenguajes y herramientas, de corte tradicional, que hemos comentado en el apartado anterior, existen unas tecnologías software de reciente aparición que, si bien no son específicas del problema de la simulación, se están mostrando como poderosos entornos que, aplicados a este campo, otorgan a la simulación una forma de programación más natural y acorde con el modelo (POO) y un impacto visual y conectividad con estaciones de trabajo mucho más potente (X-Windows).

Entre los factores que han influído en decidir la creación por nuestra parte, de una nueva aplicación especializada, debemos citar el gran coste de los sistemas profesionales existentes, su base de programación clásica, y sus amplios requerimientos de hardware. En cuanto a herramientas generales, son contados los casos en que se combinan las ventajas de la Programación Orientada a Objeto, y de X-Windows, junto con precio accesible y flexibilidad de inclusión de código propio. En vista a esta situación, nos pareció pertinente, de acuerdo con los condicionantes materiales de nuestro ámbito y deseando abrir alternativas, el desarrollo de nuestra aplicación partiendo de la potencialidad fundamental de la POO y X-Windows.

En lo que sigue, efectuaremos una breve revisión de la metodología de programación y de las herramientas que hemos empleado.

II.2.1.-PROGRAMACION ORIENTADA A OBJETO (POO).-

A la hora de simular un modelo, lo que sería un denso e inescrutable código en los lenguajes tradicionales empleados en la simulación (FORTRAN, sobre todo), queda reducido a un pedagógico programa cuando se recurre a la POO, ya que entonces, y en nuestro caso concreto, los conceptos termodinámicos quedarán asignados de una forma natural a objetos, y las relaciones a mensajes, con lo cual se obtiene una fuente de programación más natural, reconocible y fácil de exportar para utilizarlo en otras aplicaciones.

a) Historia.-

La POO es un tipo de programación que suscita gran interés desde hace poco tiempo. Su difusión y ámbito de aplicación es cada vez mayor, y su utilidad en la creación de aplicaciones es la clave de su éxito. Sin embargo, viene desde hace muchos años.

La historia de la POO se inicia con el lenguaje SIMULA, creado en Oslo en 1965 para simular procesos de tiempo discreto. SIMULA estaba basado, a su vez, en el lenguaje ALGOL-60. La finalidad de los creadores de estos lenguajes era la de preparar formas adecuadas de programación estructurada; de este esfuerzo surgieron poco después los lenguajes SIMULA-67, ALGOL-68 y PASCAL, este último de gran éxito y difusión.

No se puede hablar propiamente de la POO hasta la aparición de SMALLTALK, descendiente directo del SIMULA, en 1972, creado por Alan Kay, en el centro de investigación de Xerox Parc. La ventaja del SMALLTALK sobre SIMULA, que le dió amplia difusión y fama, fué, entre otras, la oferta de un entorno de programación muy completo, es decir, una plataforma desde la cual tener acceso a todas las herramientas de programación (un editor de texto, un compilador, una biblioteca de clases, depurador, gráficos, etc).

Otros lenguajes para la POO que se desarrollaron en la década de los 70 fueron FLAVORS, LOOPS, Object Lisp, etc., que eran versiones orientadas a objeto de LISP.

Como resultado de las investigaciones de Xerox Parc, hoy día se encuentran en el mercado SMALLTALK-80 (de la compañía Parc Place) y SMALLTALK/V (de Digitalk). Debido a su precio asequible y soportados por PC, son entornos POO de gran aceptación.

Otros entornos para la POO, también derivados de Xerox Parc, son Object Pascal (con el cual se escribió el sistema operativo del Macintosh), MacApp (herramienta para hacer aplicaciones) e Hypercard.

El lenguaje C++, de reciente aparición, fué creado en 1986 en los laboratorios Bell por Bjarne Stroustrup, y respaldado por AT&T. Tiene la ventaja de soportar al C (es decir, un código C es también C++, pero no viceversa), y por tanto la portabilidad es grande; sin embargo, requiere un esfuerzo del programador de C en adaptarse a la filosofía de POO, y así poder hacer un correcto uso del C++, es decir, sacar provecho a toda su potencialidad. Fué posteriormente ampliamente difundido por la compañía Borland International, para PC's, bajo el nombre de Turbo-C++. Otros, como la Free Software Foundation, a través del proyecto GNU, difundieron una versión para workstations, el GNU G++.

b) Detalles del lenguaje C++.-

CLASES Y OBJETOS:

Las clases son entes que darán lugar a la creación y existencia del objeto. Podemos decir que una clase no existe y no ocupa memoria en el programa, mientras que un objeto sí tiene esa existencia que nos permite comunicar y trabajar con él; no obstante, un objeto no podría crearse si no tenemos definida una clase que lo sustente.

Por ejemplo, y para ir introduciendo un código explicativo, los objetos *Madrileño* y *Cacereño* pertenecen todos ellos a la misma clase, *Español* (ver figura 2.1).

En el código fuente del programa (esto es, en el listado), primero definimos las clases, y luego, dentro de la función "main()" es donde establecemos el programa en cuanto a qué es lo que queremos hacer: crear un objeto, dar valores, imprimir resultados, etc. Normalmente, la definición de las clases se efectúa en un fichero cabecera, con la extensión .H, por ejemplo, EUROPEO.H, mientras que el resto se deja para ficheros con la extensión CPP, por ejemplo, EUROPEO.CPP.

La creación de un objeto se realiza mediante una sentencia como la siguiente:

```
Europeo Juan(26);
```

en la cual hemos creado el objeto *Juan* perteneciente a la clase *Europeo*. A efectos conceptuales, lo que "existe" ahora es la persona de nombre *Juan*, y cualquier pregunta o tarea a realizar será con este objeto, y no con la clase *Europeo*. La creación de dicho objeto ha sido mediante la llamada a una función constructora, como se verá enseguida.

MIEMBROS:

Puede concebirse una clase como una caja que contiene diversos elementos, denominados miembros de la clase: variables, funciones (también llamadas métodos), e incluso otras clases. Suele decirse que las clases encapsulan parámetros (variables) y comportamientos (funciones). Como regla general, sólo puede accederse a las variables a partir de las funciones declaradas como miembros de la clase.

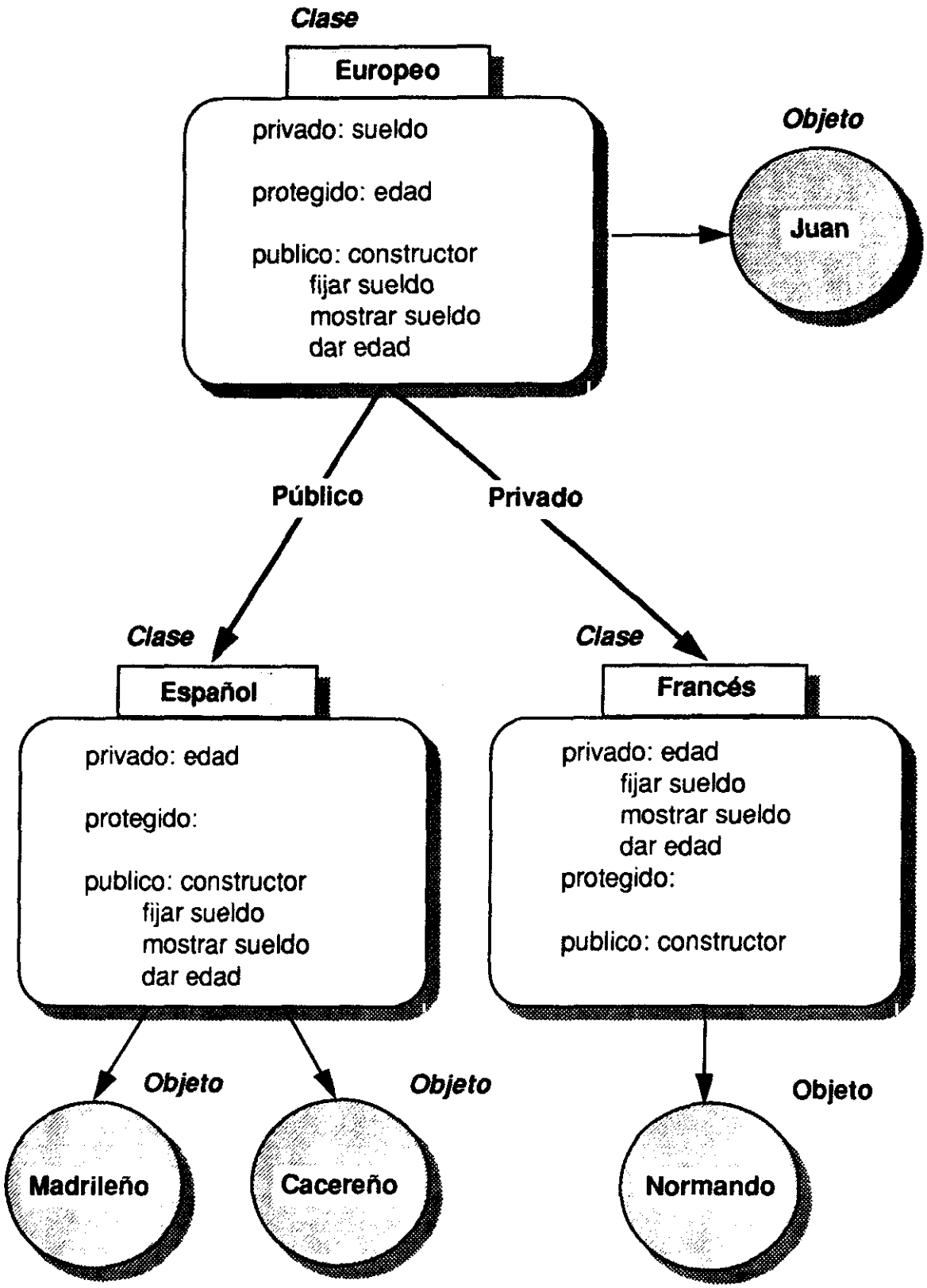

Clase
Europeo
privado: sueldo
protegido: edad
publico: constructor
fijar sueldo
mostrar sueldo
dar edad
Objeto
Juan
Público
Privado
Clase
Español
privado: edad
protegido:
publico: constructor
fijar sueldo
mostrar sueldo
dar edad
Clase
Francés
privado: edad
fijar sueldo
mostrar sueldo
dar edad
protegido:
publico: constructor
Objeto
Madrileño
Objeto
Cacereño
Objeto
Normando

Figura 2.1

Por ejemplo, la clase *Europeo* puede tener los siguientes miembros:

- Variables: sueldo, edad, nombre del padre, etc
- Funciones: fijar el sueldo, decir la edad, imprimir el sueldo, etc.
- Clases: la clase *Europeo* podría tener como miembro a la clase *Localidad*, la cual a su vez podría tener miembros como población, número de diputados, etc.

Los miembros pueden tener unas restricciones en sus accesos, dependiendo si son miembros privados, protegidos ó públicos:

- Privados: Sólo son accesibles dentro de la misma clase. Para dar valor a una variable privada hay dos formas:
  - — Mediante un constructor. Un constructor es una función pública del mismo nombre que la clase, y que permite pasar un valor. En la sentencia:

    ```
    Europeo Juan(26);
    ```

    además de crear un objeto, estamos llamando al constructor y pasando el valor 26, que se asignará a la variable privada *edad*. La ventaja de esta forma es que podemos dar valores a las variables privadas al mismo tiempo que creamos el objeto, evitando el posible olvido de dar valores mediante:
  - — Mediante una función pública. Por ejemplo, la sentencia

    ```
    Juan.fijar_sueldo(100000.);
    ```

    llama al método público *fijar_sueldo*, el cual asigna la cantidad de 100000 pta. a la variable *sueldo*. Con una misma función o con el mismo constructor podemos dar valores a cuantas variables privadas queramos.
- Protegidos: Igual que los privados, con la salvedad de que permiten la herencia, como veremos después.
- Públicos: Son accesibles desde fuera de la clase, especificando el objeto al que pertenecen. Por ejemplo, la función pública dar_edad se activa en el objeto *Juan* mediante la sentencia

```
printf( "Número de años=%d", Juan.dar_edad() );
```

en la cual se imprime mediante la función printf() el valor devuelto por la sentencia:

```
Juan.dar_edad()
```

Si hubiésemos definido la variable *sueldo* como pública, podríamos acceder a ella para dar valores sin necesidad de funciones intermedias, por ejemplo, haciendo:

```
Juan.sueldo=100000.;
```

de forma que le damos valor directamente.

HERENCIA:

Una clase puede ser derivada de otra, heredando así sus miembros. Esta herencia de miembros depende de las condiciones impuestas por la clase base (es decir, si sus miembros son privados, públicos o protegidos) y por la clase derivada (si ésta es creada privada o públicamente).

Por ejemplo, creamos las clases *Español* y *Francés* derivadas de la clase base *Europeo*; la primera la creamos públicamente y la segunda privadamente, de forma que, como se ve en la figura 2.1, cada una de estas clases tienen los mismos miembros heredados, pero con grados de protección distintos.

Ahora podemos crear los objetos *Madrileño* y *Cacereño* de la clase *Español*, y el objeto *Normando* de la clase *Francés*.

Sobre la herencia haremos varias observaciones:

- Los constructores nunca se heredan, de manera que hay que definir los constructores de las clases *Español* y *Francés*.

- Cuando una clase se deriva de otra privadamente (por defecto, éste es el caso), los miembros protegidos y públicos de la clase base pasan a ser miembros privados de la clase derivada. Cuando se crea la clase públicamente, entonces los miembros protegidos de la clase base pasan a ser privados también, pero los públicos pasan a ser públicos. Los miembros privados nunca se heredan.

- Debido a que los miembros privados no se heredan, no tiene sentido que utilicemos el método *fijar_sueldo* (público en *Español* y privado en *Francés*) en las clases derivadas, pues éstas no poseen la variable *sueldo*. Además, para poder acceder al valor de la variable *edad*, tenemos que construir un método específico para la clase *Francés* que nos dé dicho valor, pues la función *dar_edad* es privada en esta clase y por tanto inaccesible desde el programa.

- Cuando hay clases derivadas de otras, es necesario que la clase base posea un "constructor por defecto".

A continuación podemos ver el listado del código C++ que realiza el anterior planteamiento orientado a objeto.

```
/*============= EUROPEO.CPP =============*/
#include<stdio.h>
#include<conio.h>

class Europeo                          /* declaro y defino la clase base */
      {
      private:
            float sueldo;
      protected:
            int edad;
      public:
            Europeo() {}               /* constructor por defecto */
            Europeo(int n)             /* constructor */
                  {edad=n;}
            void fijar_sueldo(float x)
                  {sueldo=x;}
            float muestra_sueldo(void)
                  {return sueldo;}
            int dar_edad(void)
                  {return edad;}
      };
/*-------------------------------------------------------------------------------------------------*/
class Español:public Europeo           /* declaro clase derivada pública */
      {
      public:
            Español(int n)             /* constructor */
                  {edad=n;}
      };
/*-------------------------------------------------------------------------------------------------*/
class Frances:private Europeo          /* declaro clase derivada privada */
      {
      public:
            Frances(int n)             /* constructor */
                  {edad=n;}
            int Frances_dar_edad(void)
                  {return edad;}
      };
/*-------------------------------------------------------------------------------------------------*/
main()                                 /* cuerpo del programa principal */
      {
      Europeo Juan(26);
      Juan.fijar_sueldo(100000.);
      printf("Sueldo de Juan=%f\n",Juan.muestra_sueldo());
      printf("Edad de Juan=%d\n",Juan.dar_edad());
      Español Madrileño(30); Español Cacereño(20); Frances Normando(27);
      printf("Madrileño:edad=%d\n",Madrileño.dar_edad());
      printf("Cacereño:edad=%d\n",Cacereño.dar_edad());
      printf("Normando:edad=%d\n",Normando.dar_edad());
}
/*============== FIN ==============*/
```

II.2.2 .- Entornos Windows.-

Como propósito de potenciar el uso de las aplicaciones, fueron creados los entornos Windows, no sólo para mejorar la apariencia gráfica, sino también facilitar la comunicación de las aplicaciones entre los usuarios, establecer una normalización de formatos, etc.

Los entornos windows consideran dos aspectos: la aplicación y el entorno:

- La **aplicación**: Un programa creado para desarrollar unas tareas, se considera que es una aplicación windows cuando ha sido programado bajo unas determinadas normas y utilizando ciertas librerías para dotarlo de una apariencia característica de estos entornos, y que incluyen el uso de ventanas desplegables, interactividad con el ratón, etc.

- El **entorno**: Windows es un paquete software que se instala en la máquina, ofreciendo un entorno de trabajo para ejecutar, manipular, ordenar, etc. las aplicaciones. Ofrece herramientas y utilidades para manipular ficheros y directorios, efectuar dibujos, procesar textos, establecer comunicaciones con otros usuarios, etc. Resumidamente, Windows hace todo lo que puede hacer el ordenador, bajo un entorno de fácil uso. La instalación del entorno nos provee de unas librerías que se usarán en el caso de que programemos nuestra propia aplicación bajo norma windows.

Básicamente, hay dos potentes entornos windows que nos interesan, según la máquina opere bajo el sistema operativo DOS o bajo UNIX (por sencillez no nos referimos a otros entornos posibles, como son OS/2 o los de Apple). Para el primer caso, existe **Microsoft Windows**, creado por primera vez en 1985, y que alcanza su máxima difusión a finales de los 80 y principios de los 90, debido a la aparición de ordenadores PC con procesadores más perfeccionados (386, 486) y más rápidos. En cambio, para ordenadores y estaciones de trabajo bajo sistemas UNIX, con la misma antiguedad existe la plataforma **X-Windows**.

Los PC no pueden competir, en velocidad de procesamiento, con las estaciones de trabajo. Es por eso que el entorno X-Window es mucho más potente y ha sido desarrollado en un grado muy superior a su pariente en DOS. Su utilización cae en ámbitos más profesionales, debido a que el costo de las estaciones de trabajo no permiten adquirir una para un uso personal. Por tanto, hablar de Microsoft Windows se asocia a aplicaciones personales, ofimática, etc. y X-Windows hace referencia a un ámbito más profesional, como aplicaciones potentes, ambientes industriales, etc.

II.3.- BASE TECNOLOGICA DE NUESTRO TRABAJO.-

Como base tecnológica entendemos tanto el software utilizado como el hardware requerido para el desarrollo de nuestras aplicaciones.

Hemos desarrollado programas tanto en PC bajo DOS como en una estación de trabajo con UNIX; hemos usado distintos compiladores y utilidades gráficas. Para dar una visión de sus características, dividiremos este tema en dos apartados. En el primero, hablaremos del hardware que hemos utilizado, y en el segundo, comentaremos extensamente los paquetes software con los que hemos conseguido crear nuestras aplicaciones.

II.3.1.- HARDWARE.-

IBM PC 386

Hemos usado un PC compatible con procesador Intel 386 a 25 MHz, bajo sistema operativo MS-DOS v 4.01 para desarrollar programas escritos y compilados en C y C++ (Turbo C v2.0 de Borland y Turbo C++ v1.0 de la misma compañía). Estas aplicaciones, llamadas RANKINE.EXE y PLANTA.EXE requieren, para ser ejecutados, un PC XT/AT compatible, con al menos 640 Kb RAM. Se recomienda una tarjeta gráfica VGA color. Para la aplicación RANKINE.EXE cualquier tarjeta es suficiente, pero para PLANTA.EXE es necesario que el monitor soporte una tarjeta a color para poder identificar las numerosas curvas que representan la simulación. Por otra parte, la entidad de esta simulación hace recomendable tener un procesador lo más rápido posible, y mejor aún si va provisto de un coprocesador matemático.

SUN SPARCstation 1+

La empresa Sun Microsystems ha diseñado una arquitectura RISC, llamada arquitectura de procesador escalable (Scalable Processor ARChitecture, SPARC) para implementarla sobre la familia de workstations Sun-4.

Sobre esta máquina hemos desarrollado nuestra aplicación PLANTA, orientada a objeto y bajo entorno X-Windows. La simulación de nuestro proceso requería una mayor velocidad que la conseguida en el ya citado PLANTA.EXE, de forma que, si comparamos el rendimiento de los procesadores de nuestros ordenadores, vemos que:

CPU	MIPS (Millones de Instrucciones por Segundo)
Intel 80386	3.44
SPARC	10.69

La estación SUN Sparc1+ que utilizamos, y que podemos observar en la figura 2.2 tiene el sistema operativo SUN-OS 4.1.1, 16 Mb de memoria RAM, procesador a 25 MHz con 15.8 MIPS, disco duro interno de 600 Mb, unidad de diskette de 3.5" HD y stramer, monitor color de 19" , ratón óptico y todo tipo de puertos.

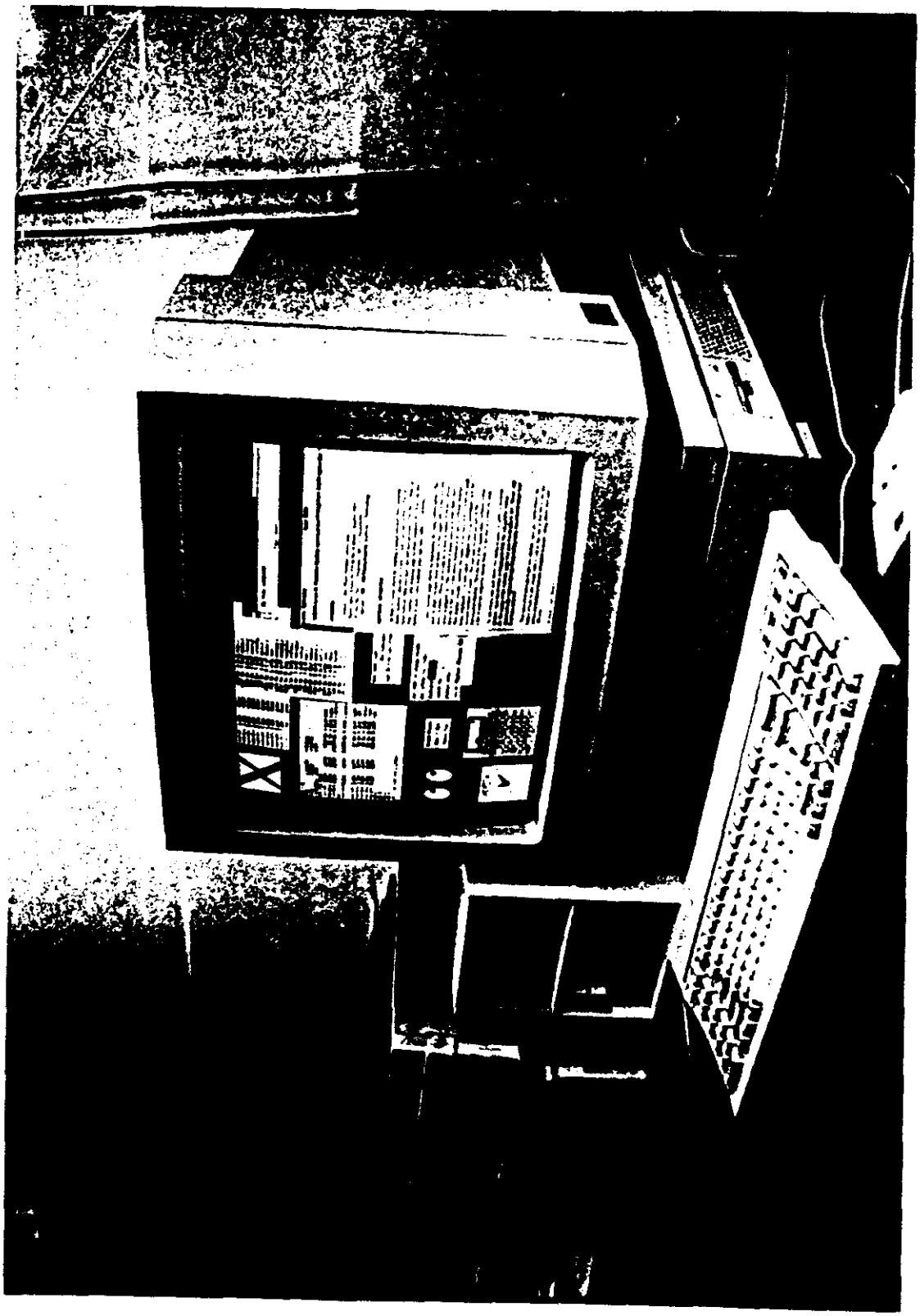


Figura 2.2

II.3.2.- SOFTWARE.-

Turbo C v2.0 - Borland Inc.

Presentado en 5 diskettes de 5.25" de baja densidad, Turbo C es un completo entorno de programación en C que incluye:

— Un entorno integrado que consta de un editor de texto, un compilador, un gestionador de ficheros, menús de opciones y configuración, etc.

— Un compilador independiente en forma de línea de comandos, que permite compilar códigos que incluyan instrucciones en lenguaje máquina.

— Ficheros cabecera y librerías C standard, más librerías gráficas propias de Borland.

Los requisitos para poder instalar y utilizar este compilador son:

— IBM PC XT/AT o compatibles, ó PS/2, con al menos 384 Kb RAM.

— Dos unidades de disco (de cualquier tamaño) o una unidad de disco y un disco duro.

— Monitor monocromo ó color.

La experiencia nos dice que es una excelente plataforma tanto para el inicio en la programación C, como para desarrollar aplicaciones de gran entidad. Es de un fácil uso debido a su entorno integrado. La librería gráfica es suficiente.

En la figura 2.3 podemos ver un aspecto de su entorno.

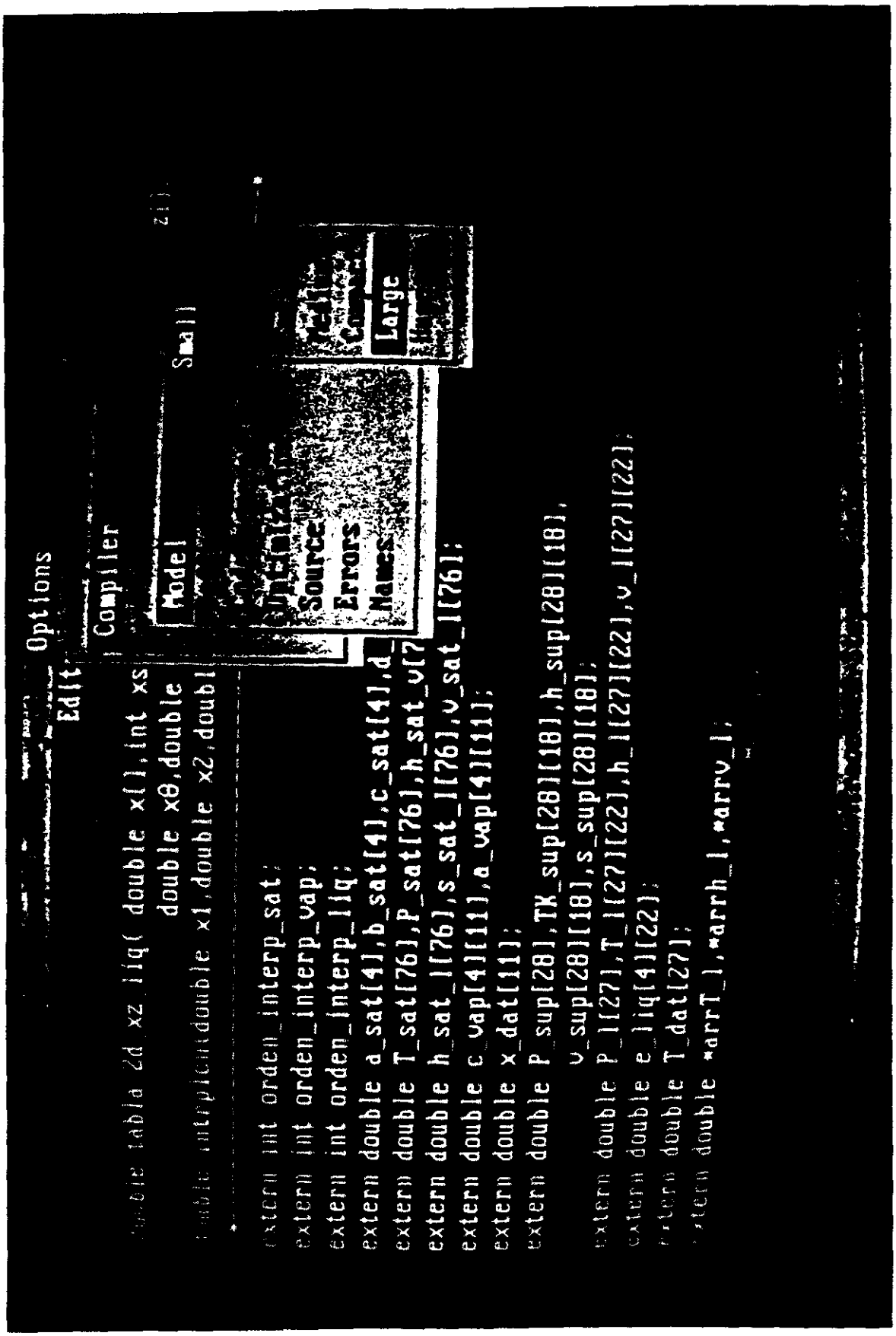

Options
Edit
Compiler
Model
Small
Source
Errors
Names
Large
extern int orden_interp_sat;
extern int orden_interp_vap;
extern int orden_interp_liq;
extern double a_sat[4],b_sat[4],c_sat[4],d_
extern double T_sat[76],P_sat[76],h_sat_v[7
extern double h_sat_l[76],s_sat_l[76],v_sat_l[76];
extern double c_vap[4][11],a_vap[4][11];
extern double x_dat[11];
extern double P_sup[28],TK_sup[28][18],h_sup[28][18],
v_sup[28][18],s_sup[28][18];
extern double P_l[27],T_l[27][22],h_l[27][22],v_l[27][22];
extern double e_liq[4][22];
extern double T_dat[27];
extern double *arrT_l,*arrh_l,*arrv_l;

Figura 2.3

Aprovechando el éxito de este compilador, se han generado multitud de librerías científicas para diversos usos. Entre éstas, hemos usado la librería "**Science & Engineering Tools for Turbo C**", de la empresa **Quinn-Curtis**. Incluye funciones estadísticas, gráficas, ajuste de datos, cálculos matriciales, etc.

Turbo C++ v2.0 - Borland Inc.

Este paquete, desarrollado por la misma compañía, ofrece tres novedades frente al anteriormente comentado Turbo C:

— Permite la programación orientada a objeto. Los códigos fuente se identifican por la extensión .CPP si utilizan las características de la POO. Cualquier código C (extensión .C) también puede ser compilado. Este paquete trae librerías de clases y abundantes ejemplos de POO. Los ficheros proyecto son distintos de los de Turbo C.

— El entorno trae, aparte del editor, compilador, etc., un depurador de errores, editor de código máquina, utilidades de ayuda a la programación, etc. Para compilar cualquier código que contenga sentencias en ensamblador no hace falta (como era el caso en Turbo C) acudir al compilador independiente, pues el compilador del entorno integrado lo hace. Se incluye un tutorial interactivo para el aprendizaje del manejo del entorno integrado.

— La presentación viene con ventanas desplegables y movibles, interactividad con ratón, potentes ayudas on-line con ejemplos (Turbo C no incluye estos ejemplos en sus ayudas), etc.

Los requisitos de hardware son los mismos que para Turbo C, si bien es recomendable, para la velocidad de compilación, un buen procesador y suficiente memoria RAM. En el caso de usar dos disqueteras (se recomienda mejor el disco duro), éstas han de ser de alta capacidad. El ratón (opcional) dá suficiente comodidad para el uso del entorno integrado. En la figura 2.4 podemos ver un aspecto de su entorno.

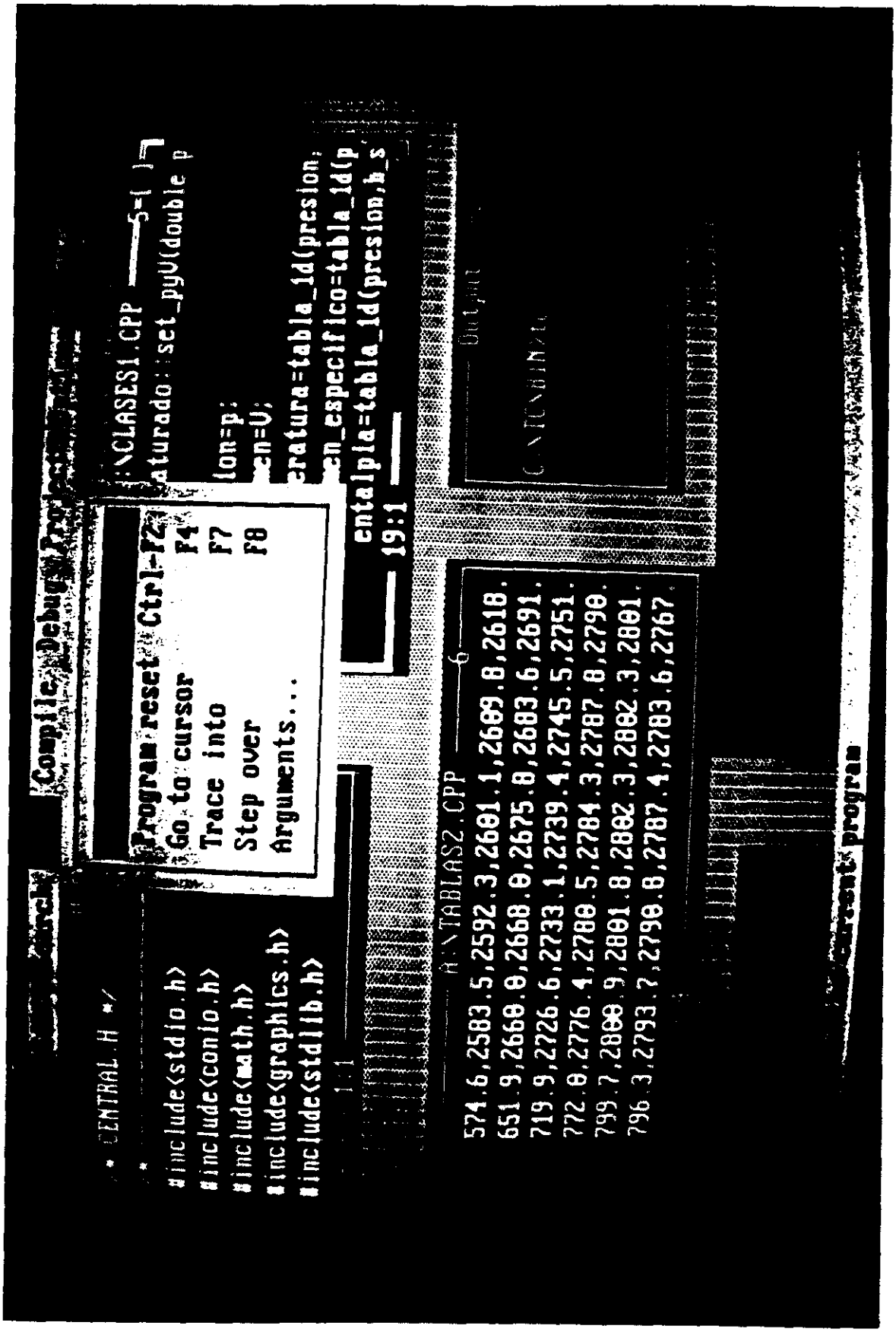

Compile Debug
Program reset Ctrl-F2
Go to cursor F4
Trace into F7
Step over F8
Arguments...
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
#include<stdlib.h>
19:1
C:\TABLAS2.CPP
574.6,2583.5,2592.3,2601.1,2609.8,2618.
651.9,2660.0,2668.0,2675.8,2683.6,2691.
719.9,2726.6,2733.1,2739.4,2745.5,2751.
772.0,2776.4,2780.5,2784.3,2787.8,2790.
799.7,2800.9,2801.8,2802.3,2802.3,2801.
796.3,2793.2,2790.8,2787.4,2783.6,2767.

Figura 2.4

GNU G++

GNU G++ es un compilador C++ instalable en un buen número de máquinas bajo diversos sistemas operativos UNIX. Hemos conseguido instalarlo tanto en la estación Sun como en un PC 386.

GNU (GNU's Not Unix) es el nombre de un sistema integrado con diverso software para UNIX (comunicaciones, compiladores de diversos lenguajes, X-Windows, todo tipo de aplicaciones y utilidades, etc.). La característica de este software es que es de dominio público, a través de la Free Software Foundation, con el fin de, no sólo difundir la copia de programas, sino promover el intercambio de aplicaciones desarrolladas por los usuarios.

G++ es una mejora de GCC (compilador C de GNU) para compilar códigos C++, y para usarlo hay que haber instalado previamente GCC (ambos, con el mismo número de versión; en nuestro caso, hemos utilizado la versión 1.39). Los ejecutables correspondientes son g++ y gcc.

Incluye una extensa librería de clases y herramientas, libg++, así como un depurador (GDB) y utilidades de código máquina.

Puede instalarse en un amplio rango de máquinas (estaciones de trabajo y PC's) que estén bajo diversos sistemas operativos UNIX: Vaxes (BSD, VMS, system V), PC 386 (ESIX, AIX), IBM PS/2 (AIX), familia SUN (Sun-Os 2,3 y 4), Alliant, Tahoe (BSD, DBX), DEC 3100 Mips, Convex C1 y C2, HP 9000, Pyramid, ISI 68000, NexT, NCR Tower 32, Altos 3068, etc.

MIT X-WINDOWS v11 R4

Creado por MIT X Consortium, hemos conseguido, a través del software GNU, las cintas conteniendo todos los códigos necesarios para la instalación y uso de la versión 11, en su realización 4, del sistema X Window. En la figura 2.5 podemos ver un aspecto de su presentación.

Hay que reseñar que no hemos podido disponer de ningún tipo de herramientas de programación para generar aplicaciones X-Windows , y hemos preferido por consideraciones de velocidad programar mediante las librerías XLib nuestras propias funciones específicas para dotar a nuestra aplicación de la interactividad con el ratón, gestión (apertura, cierre, etc.) de ventanas, diálogos, etc.

Debido a la importancia que para nosotros ha tenido este entorno, comentaremos a continuación un poco sus antecedentes y características generales, para después explicar qué límites hemos tenido en nuestra versión.

Historia.

En 1984, se planteó en el MIT (Massachusetts Institute of Technology) el siguiente problema. Tenían un buen número de workstations, incompatibles entre sí, con distintas arquitecturas y distintos sistemas operativos, adquiridas mediante donaciones o compras. Para intentar homogeneizarlas mediante una red gráfica que las abarcase a todas, el MIT formuló el Proyecto Athena, en colaboración con DEC e IBM.

La solución de este proyecto fué el diseño de una red que ejecutase aplicaciones locales, siendo posible llamar a fuentes remotas, y que operase sobre un amplio rango de estaciones de trabajo. El resultado fué X-Windows. Su primera versión para la difusión, la v10.4, estuvo lista en 1986. Posteriormente, y bajo la creación del X-Consortium, en 1988 se lanzó la versión v11, en su realización 2, puesto que la 1 era incompleta. En la actualidad se ha llegado hasta la realización 4 (probablemente aparezca ya la 5). Numerosas aplicaciones y mejoras en los interfaces gráficos de usuario, por parte de diversos equipos, han sido llevadas a cabo bajo los standards X Windows.

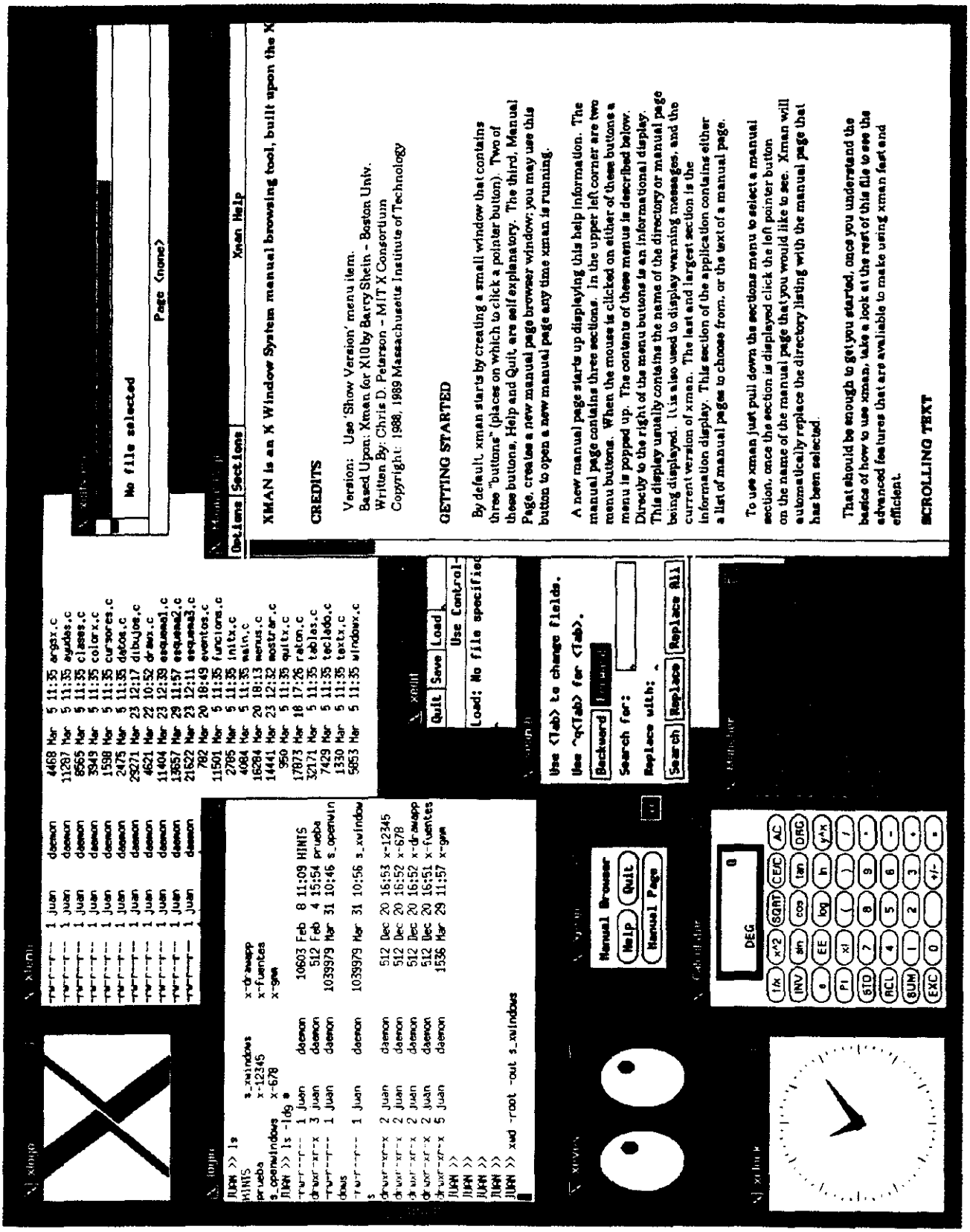

No file selected
Page <none>
Options Sections Xman Help
XMAN is an X Window System manual browsing tool, built upon the X
CREDITS
Version: Use 'Show Version' menu item.
Based Upon: Xman for X10 by Barry Shein - Boston Univ.
Written By: Chris D. Peterson - MIT X Consortium
Copyright: 1988, 1989 Massachusetts Institute of Technology
GETTING STARTED
By default, xman starts by creating a small window that contains three "buttons" (places on which to click a pointer button). Two of these buttons, Help and Quit, are self explanatory. The third, Manual Page, creates a new manual page browser window; you may use this button to open a new manual page any time xman is running.
A new manual page starts up displaying this help information. The manual page contains three sections. In the upper left corner are two menu buttons. When the mouse is clicked on either of these buttons a menu is popped up. The contents of these menus is described below. Directly to the right of the menu buttons is an informational display. This display usually contains the name of the directory or manual page being displayed. It is also used to display warning messages, and the current version of xman. The last and largest section is the information display. This section of the application contains either a list of manual pages to choose from, or the text of a manual page.
To use xman just pull down the sections menu to select a manual section, once the section is displayed click the left pointer button on the name of the manual page that you would like to see. Xman will automatically replace the directory listing with the manual page that has been selected.
That should be enough to get you started, once you understand the basics of how to use xman, take a look at the rest of this file to see the advanced features that are available to make using xman fast and efficient.
SCROLLING TEXT
Manual Browser
Help Quit
Manual Page
Quit Save Load
Use Control-
Load: No file specifie
Use <Tab> to change fields.
Use ^q<Tab> for <Tab>.
Backward
Search for:
Replace with:
Search Replace Replace All
DEG
0

Figura 2.5

Características.

El sistema X Window sostiene, básicamente, un flexible interfaz de usuario en estilo de ventanas, independiente de la máquina y adaptado para usarlo en red. Así, el interfaz de usuario sólo hace llamadas al sistema X, no al sistema operativo. También pueden crearse distintos interfaces de usuario. Gracias a todo ésto, se ha conseguido dar una misma apariencia a distintos sistemas UNIX, los cuales tenían interfaces de usuario harto diferentes entre sí.

La arquitectura del sistema X está basada de una simple relación cliente/servidor en la cual el *servidor (display server)* es el programa que controla y dibuja todas las salidas a los monitores, rastrea las entradas de clientes y actualiza las ventanas, mientras que los *clientes* son los programas de aplicación para ejecutar las tareas determinadas. Como X es un sistema en red, el cliente y el servidor no tienen que ser computados necesariamente por la misma máquina, de forma que se permiten los procesamientos distribuidos. Así, por ejemplo, una estación Sun puede ejecutar el servidor X llamando a procesar sobre un supercomputador Cray, cuyos resultados pueden verse en el monitor de la Sun.

En X, el *display* puede ser un teclado, ratón, y una o más pantallas, asociadas con una estación de trabajo. El servidor rastrea las múltiples entradas (véase la figura 2.6), permitiendo a los usuarios ejecutar diferentes clientes (aplicaciones como procesadores de textos, bases de datos, etc.). El servidor regula el tráfico entre estos programas que se ejecutan sobre sistemas locales o remotos, y el sistema local. Puede hacer:

— Permitir el acceso al display a otros clientes.

— Enviar mensajes de red.

— Interceptar otros mensajes de red de otros clientes.

— Dibujar en dos dimensiones.

— Rastrear ventanas, cursores, tipos de letra y todo contexto gráfico.

— Permitir procesar distribuidamente.

— Permitir la multitarea, si X se utiliza con un sistema operativo multitarea, como son los UNIX.

El servidor rastrea las entradas e informa a los cllientes; tales entradas son llamadas *eventos*; por ejemplo, presionar una tecla sería un evento.

Los *gestionadores de ventanas* (*windows managers*), son clientes que gestionan la localización, tamaño, aspecto, etc. de las ventanas en la pantalla. Existen varios de éstos, si bien todo sistema X viene provisto del **uwm** (Universal or Ultrix Window Manager). Otros son **awm** (Ardent Window Manager), **mwm** (Motif Window Manager) y **twm** (Tom's or Tab Window Manager).

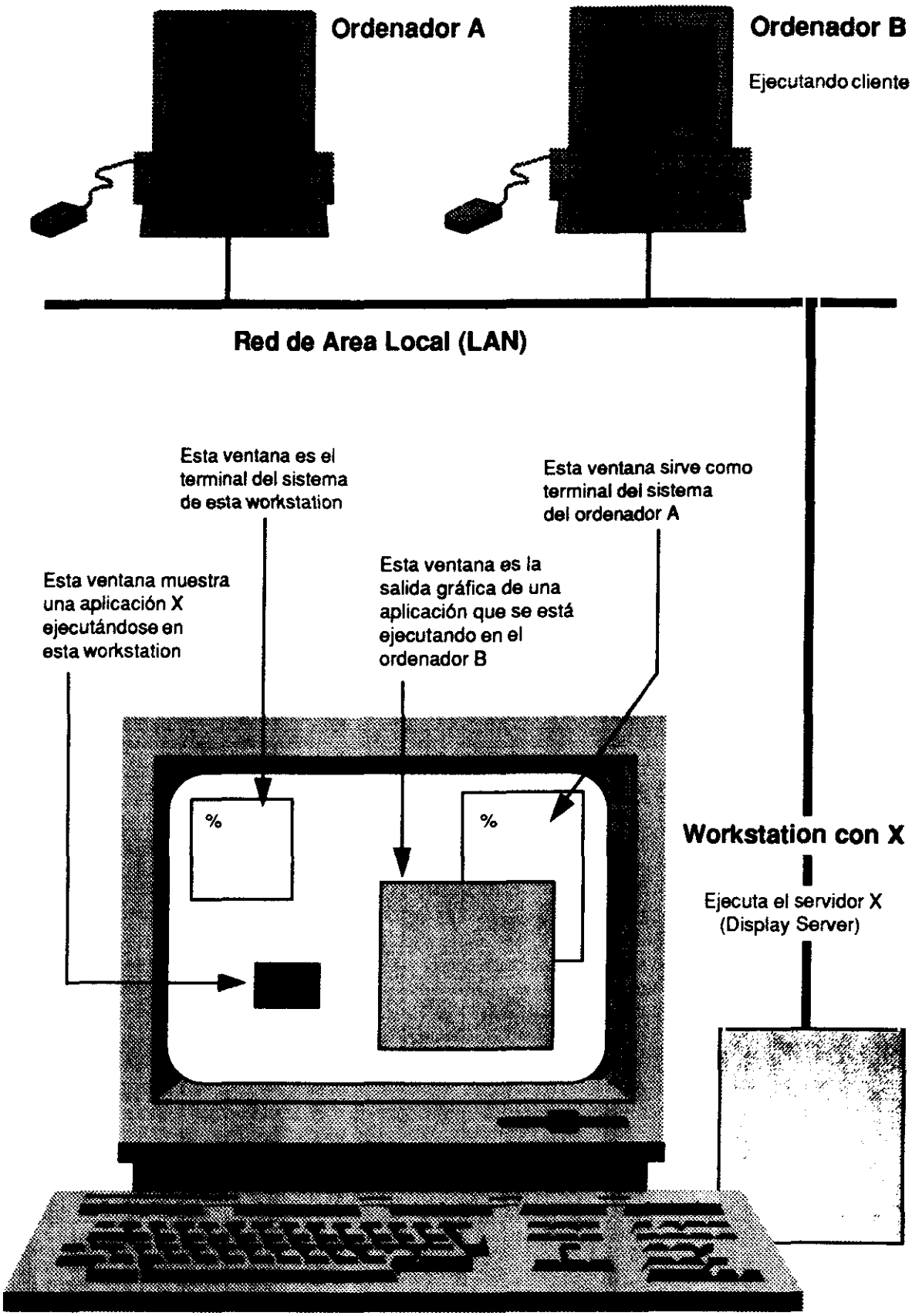

Ordenador A
Ordenador B
Ejecutando cliente
Red de Area Local (LAN)
Esta ventana es el terminal del sistema de esta workstation
Esta ventana sirve como terminal del sistema del ordenador A
Esta ventana muestra una aplicación X ejecutándose en esta workstation
Esta ventana es la salida gráfica de una aplicación que se está ejecutando en el ordenador B
%
%
Workstation con X
Ejecuta el servidor X (Display Server)

Figura 2.6

Componentes.

El sistema X Window se compone de la librería de subrutinas gráficas Xlib, el protocolo de red X, una toolkit X y varios gestionadores de ventanas (antes mencionados):

— **Xlib**. Contiene cerca de 300 funciones que equipan los requisitos del protocolo X y proveen de numerosas utilidades. Lo que hace Xlib es cubrir la llamada a una función C, como XDrawLine, con una petición del protocolo X que dibuja, en este caso, una línea. Las funciones de Xlib permiten acceder a las ventanas de distintas formas, crear, destruir, mover y dar tamaño a las mismas, dibujar líneas, polígonos, rastrear el ratón, el teclado, soportar múltiples tipos de letra, dar aplicaciones de color, etc.

— **X Toolkits**. Son librerías de rutinas que permiten programar de una manera más fácil. Así, no es necesario acudir directamente a la programación con las rutinas de Xlib para implementar nuestra aplicación, de manera que con las Toolkits podemos dotar a nuestras ventanas de botones, barras, diálogos, etc, sin necesidad de programarlos específicamente. Estas herramientas son distribuidas por compañías como Hewlett-Packard, ATT/Sun, IBM/Project Athena, DEC y Sony.

— **X Network Protocol**. El protocolo de red X define las estructuras de datos usadas para transmitir las peticiones entre clientes y servidores. Es una comunicación entre procesos basada en flujos asíncronos, en lugar de serlo mediante llamadas a procedimientos. Es una función de Xlib.

Requisitos.

Cualquier ordenador (micro, mini, super), con cualquier sistema operativo puede ejecutar X. Puede instalarse con todo o parte de su potencial. Nuestra experiencia nos dice que, sobre una estación de trabajo Sun va muy bien, pero sobre un PC 386 con sistema ESIX (UNIX system V) necesita un mínimo de 8 Mb RAM para poder ejecutar con una mínima velocidad. Normalmente se trabaja con estaciones de trabajo UNIX, si bien han aparecido recientemente productos que permiten instalar X Windows sobre un PC con sistema operativo DOS, compitiendo así en este entorno con Microsoft Windows.

Capítulo III:

SIMULACION ESTATICA DEL CICLO DE RANKINE

Capítulo III:

SIMULACION ESTATICA DEL CICLO DE RANKINE

Expondremos en este Capítulo los resultados de nuestra primera etapa de trabajo, consistente en la realización de un programa C para el estudio del ciclo de Rankine. Se pretendía ofrecer una visión didáctica del ciclo más importante en la realidad industrial de las plantas térmicas, y para ello se partió del estudio de dicho ciclo en su estado estacionario, según los tratados de ingeniería termodinámica.

Dividiremos el Capítulo en dos grandes apartados.

En el primero, estudiamos de forma teórica el ciclo de Rankine, teniendo a la vista que constituye el objetivo de la modelación estática por ordenador que vamos a realizar. Se explicarán las diversas configuraciones del ciclo (para aumentar el rendimiento), los dispositivos necesarios y los procesos que intervienen.

Dedicamos el segundo apartado a la creación de un programa de simulación basado en el modelo estático del ciclo de Rankine. En primer lugar comentaremos su concepción y su desarrollo. Posteriormente, y mediante una guía de usuario, daremos una explicación de su manejo y las posibilidades que ofrece. Este programa constituye la primera aportación de nuestro trabajo.

III.1.- EL CICLO DE RANKINE.-

III.1.1- CICLOS DE VAPOR.-

Por lo general, y a excepción de las centrales hidroeléctricas, las plantas de producción de energía eléctrica utilizan ciclos de vapor, los cuales adoptarán una forma u otra según las características propias de cada tipo de central (nuclear, de combustión, etc) o según el modo de operación particular de cada una. Debido a ello, el ciclo de vapor es un objeto de estudio fundamental. Pensemos en que de él depende, en gran medida, la seguridad en todas sus facetas, y el rendimiento energético y, por tanto, económico; ambas cosas son de un interés capital.

Estos ciclos de vapor son sustentados por una configuración de dispositivos, normalmente denominados máquinas térmicas, cuyo objetivo es el de convertir la transferencia de calor en trabajo. Para el análisis de cada ciclo habrá que tener en cuenta de qué tipo de fluido de trabajo se trata (agua, refrigerante-12, etc.), y qué configuración de dispositivos tiene.

La bibliografía en ingeniería termodinámica nos ofrece abundante información sobre los ciclos termodinámicos básicos como son los de Carnot (ideal, el de más alto rendimiento, y que se toma como referencia para los rendimientos de otros ciclos), Stirling (ciclo que usa pistones en lugar de los dispositivos normales de las plantas de vapor como son turbinas, condensadores, bombas, etc.), Ericsson (que usa turbinas y compresores fundamentalmente), Brayton (como el de Ericsson, pero con intercambiadores de calor), Rankine, etc., todos ellos de aplicación industrial con mayor o menor éxito y difusión. Sin embargo, los cursos en esta materia en el ambiente universitario son reacios a estudiarlos en profundidad y en su aplicación práctica, teniendo que recurrirse a publicaciones más específicas. El de Rankine es el más importante, sin duda; su utilización, con todo tipo de mejoras, variaciones, configuraciones, etc., es la más extendida en las centrales térmicas.

III.1.2.- CICLO DE RANKINE SIMPLE E IDEAL.-

Vamos a considerar una máquina que describe el ciclo termodinámico de Rankine en el cual, debido a una fuente caliente y otra fría, se va a producir un trabajo. El fluído de trabajo es el agua, la cual cambiará de estado y de propiedades termodinámicas a lo largo del ciclo.

Consideramos en todo momento el **régimen estacionario**, en el cual el ciclo ha alcanzado un equilibrio.

Nuestro objetivo último es el cálculo del **rendimiento**.

Este ciclo consta de cinco estados fundamentales, conectados por cinco procesos.

Como se puede ver mediante las figuras 3.1 (esquema de la configuración de la planta que sustenta el ciclo) y 3.2 (diagramas de los procesos termodinámicos que se producen), en (1) el agua se encuentra en el estado **líquido**. Se halla a una presión P_1 a la salida de la bomba, y a una relativa baja temperatura. Entra en la caldera, que es un habitáculo cerrado a presión constante, y que es calentado por el hogar (fuente caliente). Este calentamiento produce una transferencia de calor Q_h del hogar a la caldera, que será absorbido por el agua (recuérdese que estamos en condiciones ideales y no hay pérdidas de ningún tipo). En el interior de la caldera se produce el proceso (1)-->(2), isobárico. El agua incrementa pues ligeramente su volumen al aumentar su temperatura (expansión térmica). El final de este proceso se alcanza en (2), estado de **líquido saturado**, en el cual el agua ha alcanzado la temperatura de saturación T_a que corresponde a la presión de la caldera. Este líquido saturado sigue recibiendo calor, hasta vaporizarse: en (3) tenemos **vapor saturado**, a la misma temperatura de saturación T_a. El proceso (2)-->(3) se ha desarrollado a temperatura y presión constantes, produciéndose transferencias de calor debido al cambio de fase. El vapor así formado tiene una calidad del 100 %. En la caldera coexisten, pues, tres estados: líquido, líquido saturado y vapor saturado.

A la salida de la caldera se encuentra una turbina, a la cual llega el vapor saturado. Pasa a través de los álabes, perdiendo progresivamente su calidad: se expande. Así, aumenta su volumen, disminuye su presión y temperatura. Este proceso (hasta llegar al estado (4)) es isoentrópico. Se produce un trabajo útil. Al final del proceso, en (4) tenemos el estado de **vapor**, a una temperatura T_b.

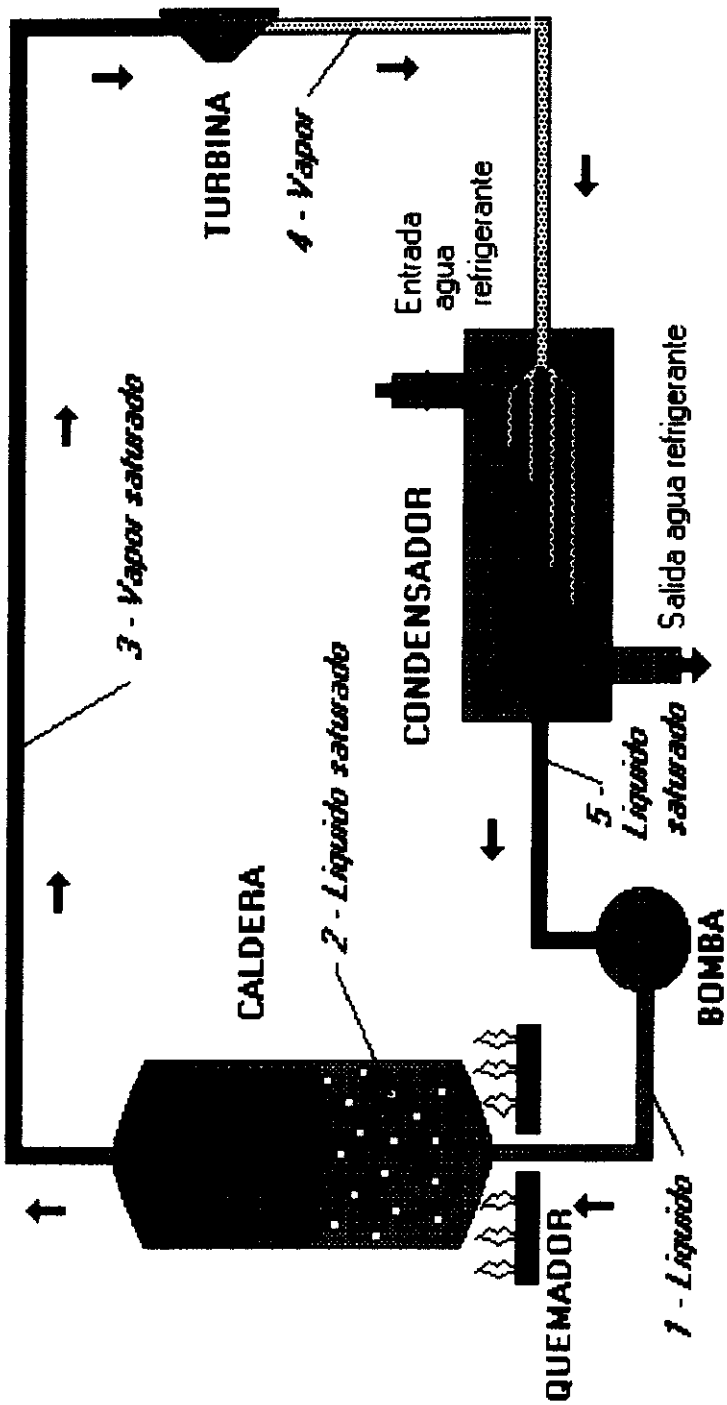

TURBINA
4 - Vapor
Entrada agua refrigerante
3 - Vapor saturado
CONDENSADOR
Salida agua refrigerante
2 - Líquido saturado
5 - Líquido saturado
CALDERA
BOMBA
QUEMADOR
1 - Líquido

Figura 3.1

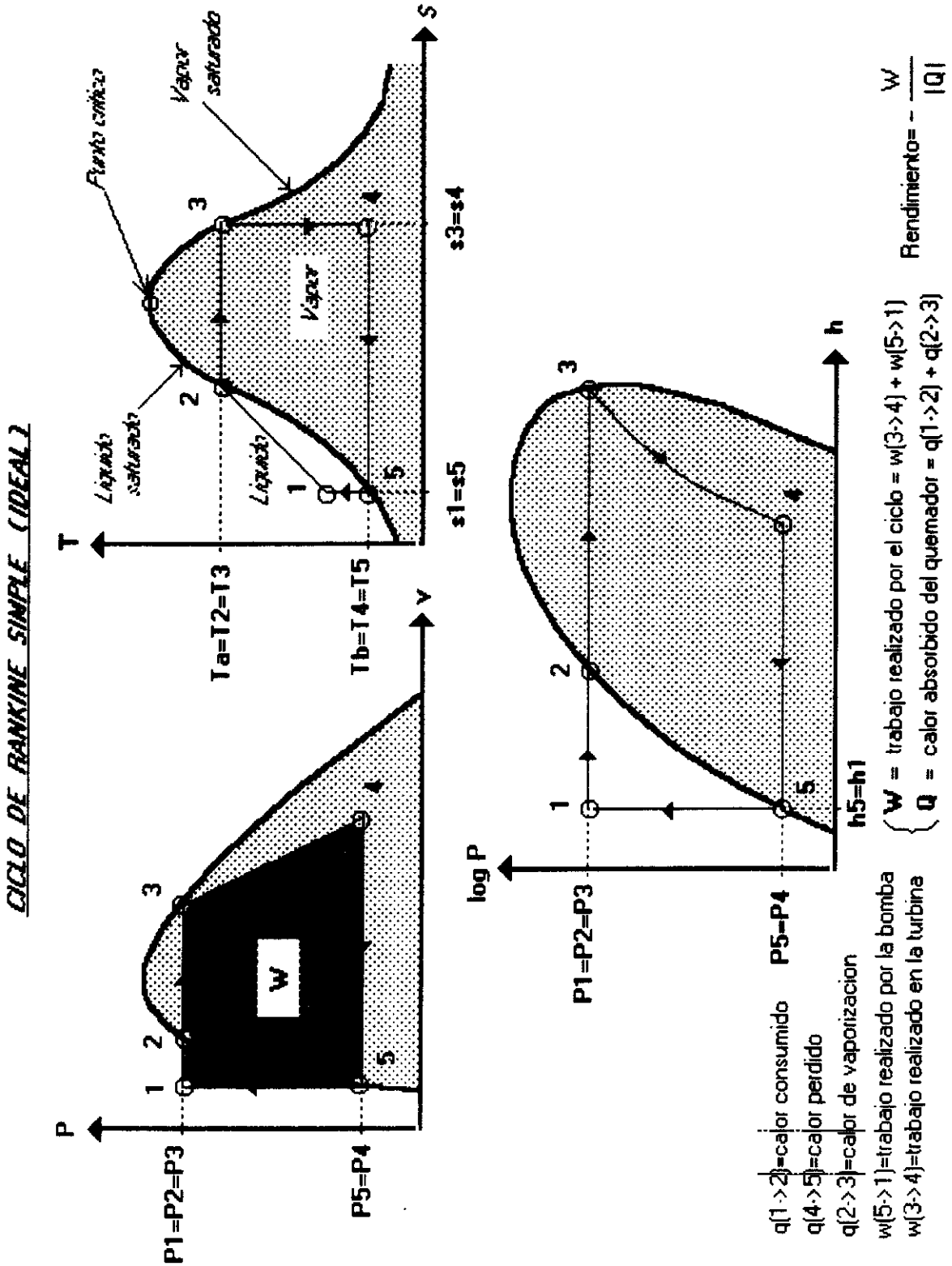

CICLO DE RANKINE SIMPLE (IDEAL)
P
T
v
s
log P
h
Punto crítico
Vapor saturado
Líquido saturado
Vapor
Líquido
W
P1=P2=P3
P5=P4
Ta=T2=T3
Tb=T4=T5
s1=s5
s3=s4
h5=h1
1
2
3
4
5
q(1->2)=calor consumido
q(4->5)=calor perdido
q(2->3)=calor de vaporizacion
w(5->1)=trabajo realizado por la bomba
w(3->4)=trabajo realizado en la turbina
W = trabajo realizado por el ciclo = w(3->4) + w(5->1)
Q = calor absorbido del quemador = q(1->2) + q(2->3)
Rendimiento= W / |Q|

Figura 3.2

Justo entonces el vapor entra en el condensador, que es refrigerado con agua fría, produciéndose la condensación del vapor. El proceso de condensación (4) --> (5) se produce a presión y temperatura (temperatura de condensación, T_b, que depende de la presión en el condensador) constantes. Hemos tomado como estado de vapor (4) aquél estado último de vapor a partir del cual se produce la condensación, de ahí que digamos que el estado (4) está a T_b. Así, el estado (5) es el de **líquido saturado**. Durante la condensación se ha producido una pérdida de calor Q_1, que es absorbida por el agua refrigerante.

A continuación, nuestro líquido de trabajo entra en una bomba, la cual consume energía para elevar la su presión hasta llevarlo de nuevo al estado (1) de **líquido**. El proceso (5)-->(1) es isoentrópico, produciéndose una despreciable variación de volumen debido a la difícil compresibilidad de este estado.

Cálculo del rendimiento.

Pretendemos calcular el rendimiento del ciclo de vapor, que no es lo mismo que el rendimiento de la planta, pues para el primero, el rendimiento es la relación entre el trabajo realizado por el fluido y el calor absorbido por el mismo, mientras que para la planta consideraríamos el gasto energético en combustible y la energía eléctrica producida.

Datos conocidos:

T_a = Temperatura de saturación.

T_b = Temperatura de condensación.

Por tanto:

$T_2 = T_3 = T_a$ $\qquad$ $x_2 = x_5 = 0$

$T_5 = T_4 = T_b$ $\qquad$ $x_3 = 1$

$P_4 = P_5$ $\qquad$ $s_3 = s_4$

$P_3 = P_2 = P_1$ $\qquad$ $s_1 = s_5$

$v_1 = v_5$

Símbolos y unidades empleados:

Temperatura:	T	ºK
Presión:	P	kPa
Volumen específico:	v	m^3/kg
Entropía:	s	kJ/(kg.ºK)
Entalpía:	h	kJ/kg
Calidad de vapor:	x	sin dimensión
Trabajo:	W	kJ/kg
Calor:	Q	kJ/kg

$$\text{Rendimiento} = R_{s,s} = \frac{\text{trabajo realizado}}{\text{calor absorbido}} = -\frac{{}_3W_4 + {}_5W_1}{{}_1Q_3}$$

- ${}_3W_4 = -\int_4^3 v.dP = \int_3^4 v.dP = \int_3^4 dh = h_4 - h_3$ (puesto que $dh = T.ds + v.dP$, $s_3 = s_4$, $ds = 0$)

- ${}_1Q_3 = {}_1Q_2 + {}_2Q_3$ $\begin{cases} {}_2Q_3 = T_2.(s_3 - s_2) & \text{puesto que } dQ = T.ds \\ {}_1Q_2 = -\int_2^1 v.dP = \int_1^2 v.dP = \int_1^2 dh = h_2 - h_1 \end{cases}$

- ${}_5W_1 = \int_5^1 v.dP = v_5.(P_1 - P_5)$ (pues v es constante)

Por tanto, se llega a:

$$R = -\frac{h_4 - h_3 + v_5.(P_1 - P_5)}{\left|(h_2 - h_1) + T_2.(s_3 - s_2)\right|}$$

Para calcular el rendimiento hemos de conocer antes algunos estados. Para ello, es imprescindible acudir a tablas termodinámicas. Para el caso de un programa informático que calcule el rendimiento, es necesario computar dichas tablas de alguna forma. Se hizo ésto, y como resultado creamos las siguientes tablas:

- **liqsat(T)**; Para un T dado, obtenemos P, v, s y h del estado **líquido saturado.**
- **vapsat(T)**; Para un T dado, obtenemos P, v, s y h del estado **vapor saturado.**
- **vapor(T,s)**; Para un T y un s dados, obtenemos x, h y v del estado de **vapor.**

Se supone que los datos de entrada que conocemos son T_a y T_b; por tanto, conocemos también T_2, T_3, T_4 y T_5, además de x_2, x_3 y x_5. A partir de este conocimiento, y usando las tablas, podemos calcular otros estados. Los pasos a seguir son éstos:

- Mediante liqsat(T_a) obtengo P_2, v_2, s_2 y h_2.
- Mediante liqsat(T_b) obtengo P_5, v_5, s_5 y h_5.
- $P_1 = P_2$, $P_3 = P_2$, $P_4 = P_5$, $s_1 = s_2$.
- $h_1 = h_5 + v_5.(P_1 - P_5)$, luego: $h_1 = h_5 + v_5.(P_1 - P_5)$
- Mediante vapsat(T_3) obtengo v_3, s_3 y h_3.
- $s_4 = s_3$.
- Mediante vapor(T_4, s_4) obtengo x_4, h_4 y v_4

Entonces, los intercambios energéticos son:

$$_1Q_2 = h_2 - h_1 > 0 \qquad _3W_4 = h_4 - h_3 < 0$$

$$_2Q_3 = h_3 - h_2 > 0 \qquad _5W_1 = h_1 - h_5 > 0$$

$$_4Q_5 = h_5 - h_4 < 0$$

Comentarios: La suma de $_5W_1$, $_1Q_2$ y $_2Q_3$ representa la energía consumida por el agua en el ciclo; $_3W_4$ es la energía útil cedida, aprovechable, y $_4Q_5$ representa la pérdida inevitable de energía.

III.1.3.- CICLO DE RANKINE SIMPLE Y REAL.-

El anterior ciclo es una aproximación ideal al ciclo de Carnot. Sin embargo, hemos de considerar que en el ciclo real se han de tener en cuenta las irreversibilidades en los procesos de expansión del gas en la turbina y compresión del líquido en la bomba. El efecto de estas irreversibilidades puede implementarse mediante la definición de los rendimientos de la turbina y la bomba.

Se dice que la turbina y la bomba tienen unos rendimientos en % en comparación con el caso ideal (100%), definiéndose por tanto de la siguiente forma:

- Rendimiento de la turbina: $R_t = \frac{{}_3W_{4a}}{{}_3W_{4s}}$

- Rendimiento de la bomba: $R_p = \frac{{}_5W_{1s}}{{}_5W_{1a}}$

donde, y a partir de ahora, emplearemos los subíndices 'a' y 's' para:

a ------> caso <u>real</u> **s** ------> caso <u>ideal</u>

Los efectos producidos por las irreversibilidades (s_{4a} no es igual qu s_3, y s_5 es distinto que s_{1a}) se aprecian en las gráficas representadas en la figura 3.3.

Cálculo del rendimiento:

$$\text{Rendimiento simple y real} = R_{s,a} = -\frac{{}_3W_{4a}+{}_5W_{1a}}{|{}_{1a}Q_3|}$$

- Datos conocidos: T_a, T_b, R_t, R_p.
- $T_2=T_a$, $T_3=T_a$, $T_5=T_b$, $T_{4a}=T_{4s}=T_b$.
- Mediante liqsat(T_2) obtengo P_2, v_2, s_2 y h_2.
- Mediante liqsat(T_5) obtengo P_5, v_5, s_5 y h_5.
- $P_{1a}=P_{1s}=P_2=P_3$ $P_5=P_{4a}=P_{4s}$.
- $x_3=0$, $x_7=0$, $x_4=0$.

- $s_{1s}=s_5$.
- Mediante vapsat(T_3) obtengo v_3, s_3 y h_3.
- $s_3=s_{4s}$.
- $h_1=h_5+v_5.(P_1-P_5)$, luego $h_1=h_7+v_7.(P_1-P_7)$.
- Mediante vapor(T_{4s},s_{4s}) obtengo x_{4s}, h_{4s} y v_{4s}.
- $\left.\begin{array}{r} {}_3W_{4s}=h_{4s}-h_3 \\ {}_3W_{4a}=h_{4a}-h_3=R_t \cdot {}_3W_{4s} \end{array}\right\} \Longrightarrow h_{4a}=h_3+(h_{4s}-h_3).\frac{R_t}{100}$
- $\left.\begin{array}{r} {}_5W_{1s}=h_{1s}-h_5=R_p \cdot {}_5W_{1a} \\ {}_5W_{1a}=h_{1a}-h_5 \end{array}\right\} \Longrightarrow h_{1a}=h_5+{}_5W_{1a}=h_5+(h_{1s}-h_5) \cdot \frac{100}{R_p}$

Los intercambios energéticos son:

Energía útil cedida: ${}_3W_{4a}=h_{4a}-h_3<0$

Energía consumida por el agua: ${}_5W_{1a}=h_{1a}-h_5>0$

${}_{1a}Q_2=h_2-h_{1a}>0$

${}_2Q_3=h_3-h_2>0$

Energía perdida: ${}_{4a}Q_5=h_5-h_{4a}<0$

Comentarios: Haciendo cuentas, se puede comprobar que $R_{s,a} < R_{s,s}$. Esta reducción del rendimiento se debe casi todo a ${}_3W_{4a}$, o sea, a la irreversibilidad de la turbina. También aumenta ${}_5W_{1a}$ pues hemos de suministrar más trabajo por la bomba al líquido saturado, elevando la entalpía del líquido, y por tanto el calor necesario para saturarlo en la caldera va a ser menor: luego la energía neta añadida (cedida) al ciclo es mayor en el caso real y menor en el ideal.

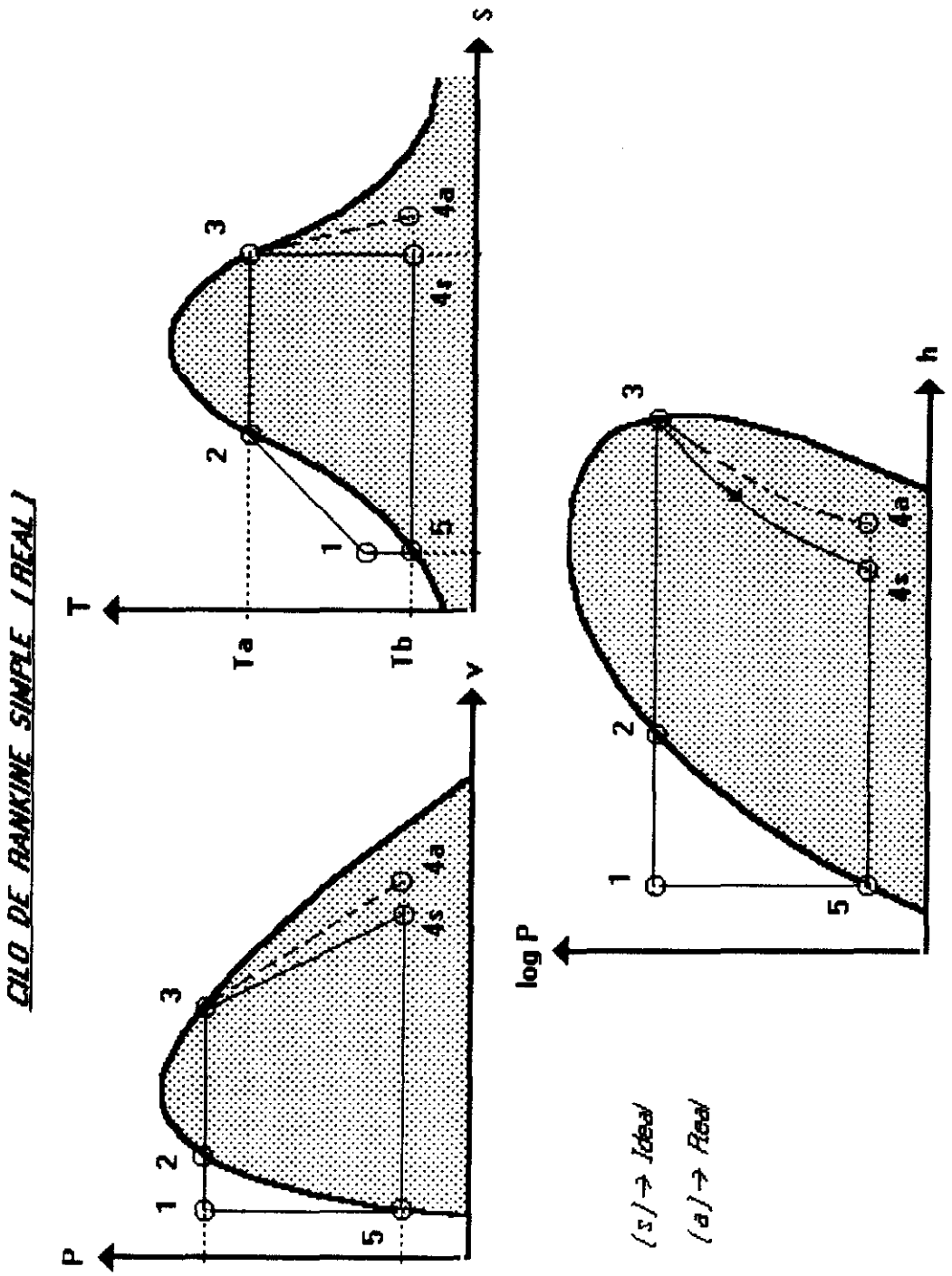

CICLO DE RANKINE SIMPLE (REAL)
T
S
Ta
Tb
1
2
3
4a
4s
5
P
v
log P
h
(s) → Ideal
(a) → Real

Figura 3.3

III.2.3.- CICLO DE RANKINE SUPERCALENTADO.-

El rendimiento del ciclo de Rankine simple es pequeño, incluso para el caso ideal. Una forma de aumentar este rendimiento es mediante la introducción de un nuevo elemento: el supercalentador. La configuración de la planta cambia, y por tanto el ciclo termodinámico también. En las figuras 3.4 (esquema de la planta) y 3.5 (diagramas termodinámicos) podemos observar este nuevo ciclo, llamado supercalentado.

Un supercalentador es un habitáculo donde entra el vapor saturado que sale de la caldera. Allí volverá a ser calentado, adquiriendo un nuevo estado, que es el de **vapor supercalentado**. Para tener las propiedades termodinámicas de este nuevo estado, es necesario recurrir a las tablas. Mediante ellas, y elaborando algoritmos de interpolación, creamos las siguientes funciones:

- **vapsupercal(T,P)**;Para un valor de T y P, obtenemos h y s del vapor en estado supercalentado.
- **vapsupercal2(P,s)**;Para un valor de P y s, obtenemos T y h del agua en estado de vapor supercalentado.

El supercalentador se encuentra a una presión determinada, que es la que va a fijar la temperatura T_{sc} que adquiere el fluido en su interior. Así, el proceso 3->4 es un proceso a presión constante, en el cual aumenta la temperatura, volumen, entalpía y entropía del gas.

Cálculo del rendimiento:

$$R_{sup,s} = -\frac{{}_{5s}W_4 + {}_{6}W_{1s}}{\left| {}_{1s}Q_2 + {}_{2}Q_3 + {}_{3}Q_4 \right|} = -\frac{h_{5s} - h_4 + h_{1s} - h_6}{\left| h_4 - h_{1s} \right|}$$

$$R_{sup,a} = -\frac{{}_{5a}W_4 + {}_{6}W_{1a}}{\left| {}_{1a}Q_2 + {}_{2}Q_3 + {}_{3}Q_4 \right|} = -\frac{h_{5a} - h_4 + h_{1a} - h_6}{\left| h_4 - h_{1a} \right|}$$

Para calcular los estados, seguimos este orden:

- Datos conocidos: T_a, T_b, R_t, R_p, T_{sc}.
- $T_2 = T_3 = T_a$ $T_{5s} = T_{5a} = T_6 = T_b$.
- $x_2 = x_6 = 0$ $x_3 = 1$.
- Mediante liqsat(T_2) obtengo P_2, v_2, s_2 y h_2.
- Mediante liqsat(T_6) obtengo P_6, v_6, s_6 y h_6.
- $P_{1s} = P_{1a} = P_2 = P_3 = P_4$ $P_{5s} = P_{5a} = P_6$.
- $s_{1s} = s_6$.
- $h_{1s}=h_6+v_6.(P_{1s}-P_6)$
- $h_{1a}=h_6+(h_{1s}-h_6)\frac{100}{R_p}$
- Mediante vapsat(T_3) obtengo v_3, s_3 y h_3.
- $T_4 = T_{sc}$
- Mediante vapsupercal(T_4,P_4) obtengo h_4 y s_4.
- $s_4 = s_{5s}$
- Mediante vapor(T_{5s},s_{5s}) obtengo x_{5s}, h_{5s} y v_{5s}.
- $h_{5a}=h_4+(h_{5s}-h_4).\frac{R_t}{100}$

Obviamente, $R_{sup,a} < R_{sup,s}$.

Por otra parte, para los mismos valores de T_a, T_b, R_t y R_p se obtiene que $R_{sup} > R_s$; es decir, hemos logrado aumentar el rendimiento con la introducción de un supercalentador.

Las transferencias energéticas son:

Energía útil cedida:	$_4W_5=h_5-h_4$	<0
Energía consumida por el agua:	$_6W_1=h_6-h_1$	>0
	$_1Q_2=h_2-h_1$	>0
	$_2Q_3=h_3-h_2$	>0
	$_3Q_4=h_4-h_3$	>0
Energía perdida:	$_5Q_6=h_6-h_5$	<0

Comentarios: El incremento del rendimiento en el ciclo supercalentado es pequeño, pero ahora tenemos una mayor potencia de salida en la turbina, $_4W_5$, respecto al ciclo simple; la causa es que el vapor que entra ahora en la turbina tiene mayor entalpía. Por otro lado, la calidad del vapor que sale de la turbina es ahora mayor.

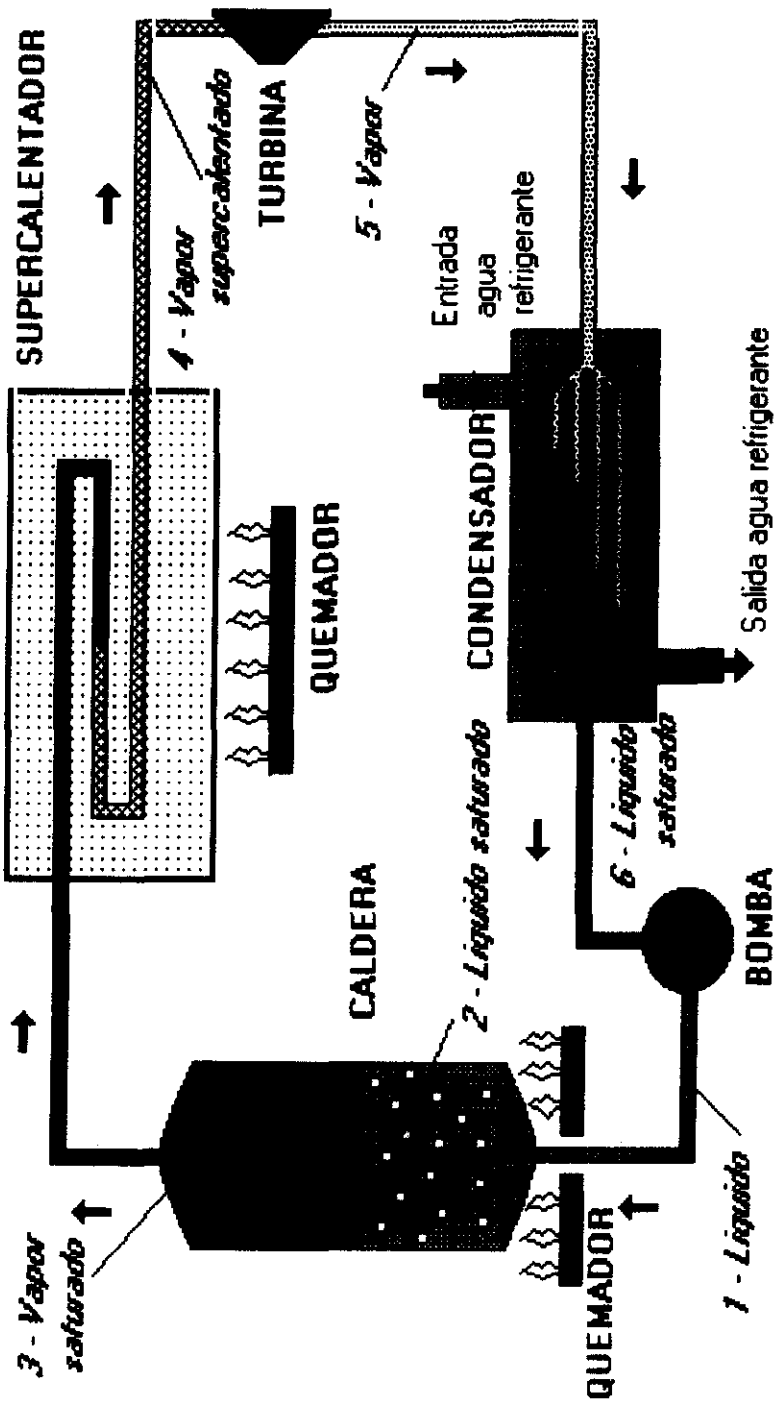

SUPERCALENTADOR
TURBINA
4 - Vapor sobrecalentado
5 - Vapor
Entrada agua refrigerante
QUEMADOR
CONDENSADOR
Salida agua refrigerante
CALDERA
2 - Líquido saturado
6 - Líquido saturado
BOMBA
3 - Vapor saturado
QUEMADOR
1 - Líquido

Figura 3.4

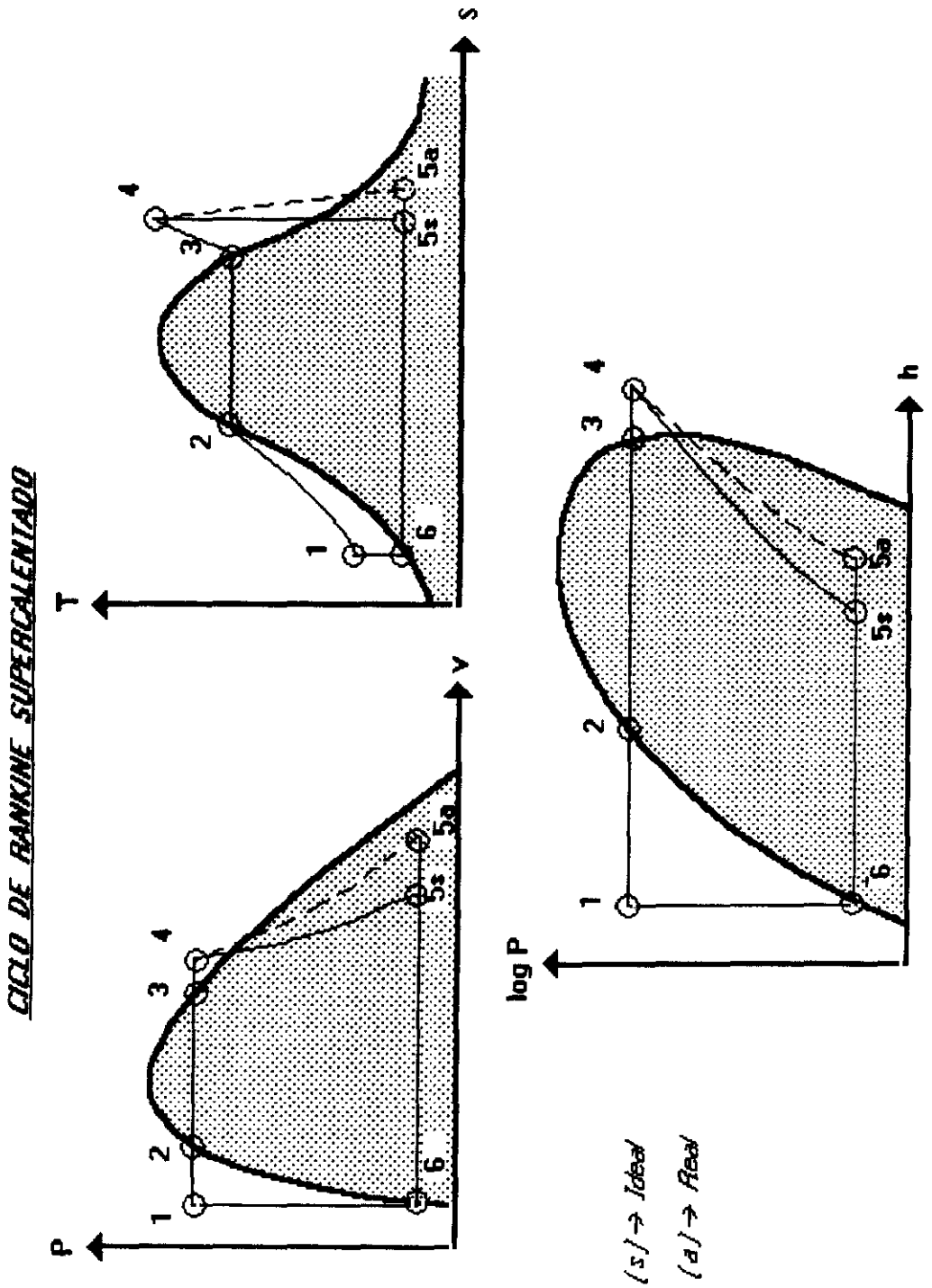

CICLO DE RANKINE SUPERCALENTADO
P
T
v
s
log P
h
1
2
3
4
5s
5a
6
(s) → Ideal
(a) → Real

Figura 3.5

III.2.4.- CICLO DE RANKINE RECALENTADO.-

Cambiamos de nuevo la configuración de la planta para intentar aumentar el rendimiento del ciclo. Esta nueva configuración podemos apreciarla en la figura 3.6, y los diagramas termodinámicos correspondientes en la figura 3.7.

El vapor saturado, (3), que sale de la caldera, entra en el supercalentador. A la salida del mismo, su estado (4) es de vapor supercalentado a alta presión. Lo pasamos por una turbina de alta presión, de manera que el vapor se enfría (5) y disminuye su presión hasta el valor Ph, que va a ser, junto con Ta, Tb y Tsc, un parámetro de control. Como lo que queremos es aumentar el rendimiento, pensemos que si logramos aumentar su volumen para pasarlo nuevamente por otra turbina, estaremos sacando más trabajo y por tanto mejorando el rendimiento. Esto debe ser, además, sin gastar más energía. Por tanto, aprovechamos la presencia del supercalentador para volver a hacer pasar el gas por él; entonces se calienta, y tenemos en (5') un vapor supercalentado a menor presión que en (4), pues ya perdió presión en la turbina.

Después, hacemos pasar ahora el gas a través de una turbina de baja presión, extrayendo nuevamente trabajo; a la salida, en (5") tenemos ya vapor a Tb, y el ciclo continúa como en los anteriores casos.

Cálculo del rendimiento:

$$R_{rec,s} = -\frac{{}_4W_{5s} + {}_{5'}W_{5''s} + {}_6W_{1s}}{\left| {}_{1s}Q_2 + {}_2Q_3 + {}_3Q_4 + {}_{5s}Q_{5'} \right|} = -\frac{h_{5s} + h_{5''s} + h_{1s} - h_4 - h_{5'} - h_6}{\left| h_4 + h_{5'} - h_{1s} - h_{5s} \right|}$$

$$R_{rec,a} = -\frac{{}_4W_{5a} + {}_{5'}W_{5''a} + {}_6W_{1a}}{\left| {}_{1a}Q_2 + {}_2Q_3 + {}_3Q_4 + {}_{5a}Q_{5'} \right|} = -\frac{h_{5a} + h_{5''a} + h_{1a} - h_4 - h_{5'} - h_6}{\left| h_4 + h_{5'} - h_{1a} - h_{5a} \right|}$$

Para calcular los estados, seguimos este orden:

- Datos conocidos: T_a, T_b, R_t, R_p, T_{sc}, P_h.
- $T_2 = T_3 = T_a$ $T_{5"s} = T_{5"a} = T_6 = T_b$.
- $T_4 = T_{5'} = T_{sc}$.
- $x_2 = x_6 = 0$ $x_3 = 1$.
- Mediante liqsat(T_2) obtengo P_2, v_2, s_2 y h_2.
- Mediante liqsat(T_6) obtengo P_6, v_6, s_6 y h_6.
- $P_{1s} = P_{1a} = P_2 = P_3 = P_4$ $P_{5"s} = P_{5"a} = P_6$.
- $P_{5s} = P_{5a} = P_{5'} = P_h$.
- $s_{1s} = s_6$.
- Mediante vapsat(T_3) obtengo v_3, s_3 y h_3.
- $h_{1s}=h_6+v_6.(P_{1s}-P_6)$
- $h_{1a}=h_6+(h_{1s}-h_6)\frac{100}{R_p}$
- Mediante vapsupercal(T_4,P_4) obtengo h_4 y s_4.
- $s_4 = s_{5s}$
- Mediante vapsupercal2(P_{5s},s_{5s}) obtengo T_{5s} y h_{5s}.
- $h_{5a}=h_4+(h_{5s}-h_4).\frac{R_t}{100}$
- Mediante vapsupercal($T_{5'}$,$P_{5'}$) obtengo $h_{5'}$ y $s_{5'}$.
- $s_{5'} = s_{5"s}$
- Mediante vapor($T_{5"s}$,$s_{5"s}$) obtengo $x_{5"s}$, $h_{5"s}$ y $v_{5"s}$.
- $h_{5"a}=h_{5'}+(h_{5"s}-h_{5'}).\frac{R_t}{100}$

Las transferencias energéticas son:

Energía útil cedida:	$_4W_5=h_5-h_4$ <0
Energía consumida por el agua:	$_6W_1=h_6-h_1$ >0
	$_1Q_2=h_2-h_1$ >0
	$_2Q_3=h_3-h_2$ >0
	$_3Q_4=h_4-h_3$ >0
	$_5Q_{5'}=h_{5'}-h_5$ >0
	$_{5'}Q_{5''}=h_{5''}-h_{5'}$ >0
Energía perdida:	$_{5''}Q_6=h_6-h_{5''}$ <0

Comentario: Se puede comprobar que el rendimiento ha subido respecto al del ciclo supercalentado.

Una ventaja del ciclo recalentado es que pueden usarse turbinas en etapas, operando cada una en un menor rango de presiones; así, es más fácil diseñar una turbina para que trabaje en un rango determinado de presiones que una que tenga que soportar todo el proceso de expansión.

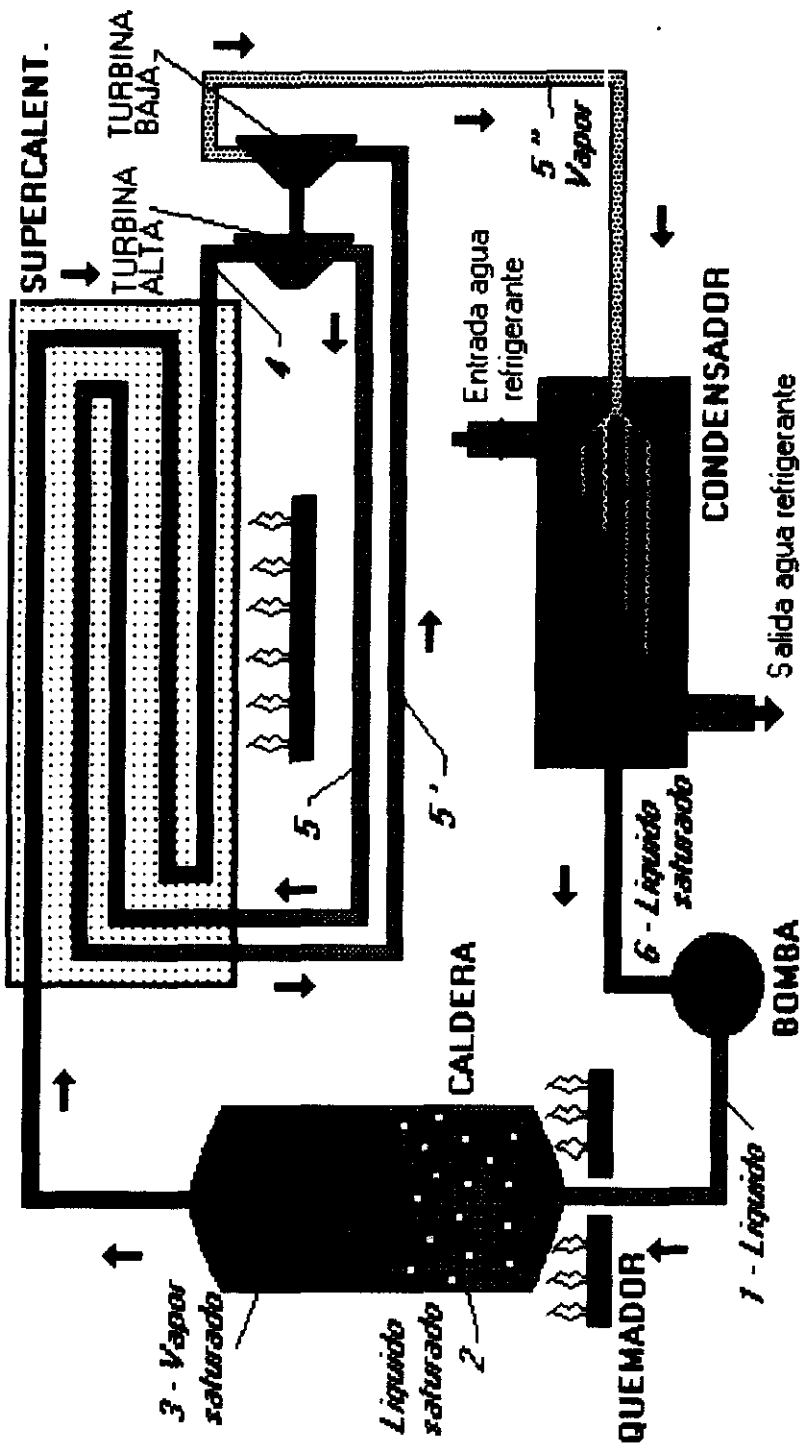

SUPERCALENT.
TURBINA BAJA
TURBINA ALTA
4
5
5'
5" Vapor
Entrada agua refrigerante
CONDENSADOR
Salida agua refrigerante
6 - Líquido saturado
BOMBA
CALDERA
1 - Líquido
3 - Vapor saturado
Líquido saturado
2
QUEMADOR

Figura 3.6

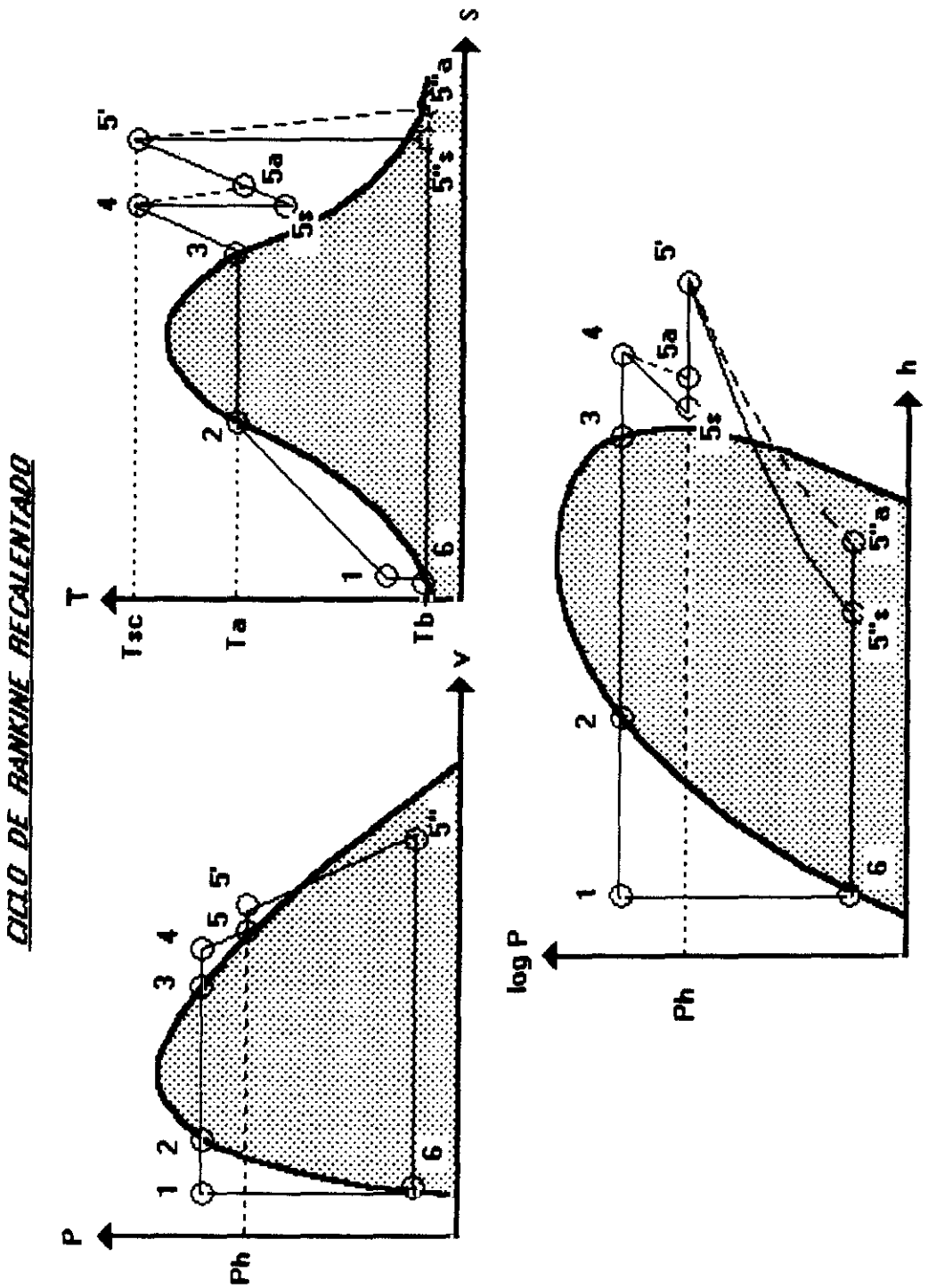

CICLO DE RANKINE RECALENTADO
T
S
Tsc
Ta
Tb
P
v
Ph
log P
h
1
2
3
4
5
5'
5a
5s
5"
5"a
5"s
6

Figura 3.7

III.2.- EL PROGRAMA RANKINE.EXE.-

III.2.1.- CONCEPCION Y DESARROLLO.-

a) CONCEPCION.

Como primera aproximación a la problemática real, lo que pretendemos es que, antes de modelar un sistema complejo y de comportamientos muy variados, tengamos unos conocimientos básicos de cómo opera la planta, los órdenes de magnitud de algunas variables, sus rendimientos, sus respuestas estacionarias, etc. Todo ello nos dará una comprensión de los fenómenos termodinámicos, muy útil a la hora de hacer una modelación que incluya transitorios, controles, comunicación interactiva, etc.

Para ello, vamos a considerar las siguientes acotaciones:

- Vamos a hacer una modelación basada en un estudio **estacionario**. Por tanto, suponemos que, tras las maniobras adecuadas, se ha logrado que la planta describa un ciclo termodinámico estable y concreto.

- Consideraremos que la planta describe el ciclo de **Rankine**, estudiando además sus diversas configuraciones (simple, supercalentado, recalentado). Tendremos unos parámetros que podremos variar y ver su efecto. Estos parámetros que podemos elegir dependiendo de la configuración concreta del ciclo son:

  — Temperatura de saturación en la caldera.

  — Temperatura de condensación en el condensador.

  — Rendimientos de la turbina y de la bomba.

  — Temperatura en el supercalentador.

  — Presión a la salida de la turbina de altas presiones.

La información que vamos a sacar de esta etapa principal de la modelación va a ser la siguiente:

— Rendimientos del ciclo de vapor. No es lo mismo el rendimiento del ciclo de vapor que el de la central, pues en el primero consideramos la energía consumida y cedida por el fluído de trabajo (vapor de agua), mientras que en la central nos interesa el combustible gastado frente a la potencia eléctrica generada. Podremos ver de qué manera el rendimiento aumenta o disminuye para cada tipo de ciclo o para valores distintos de estos parámetros.

— Intercambios de calor en las distintas partes del ciclo, su ponderación, porcentajes, etc., y efectos ante el cambio de valor inicial de los parámetros.

— Valores de las propiedades termodinámicas (temperatura, presión, entalpía, densidad, etc.) de los distintos estados del agua a lo largo del ciclo.

b) DESARROLLO.

Usaremos el lenguaje C para hacer un programa que nos calcule todos estos requisitos. La programación orientada a objeto no es esencial ahora pues no se trata de construir un modelo que nos permita futura re-usabilidad y que sea deformable a nuestros deseos con el fin de modelar lo más cerca posible del mundo real. Nuestro interés se centra en sacar partido a la teoría termodinámica y adquirir conocimientos y resultados de utilidad para futuros trabajos.

El programa que hemos realizado se llama RANKINE.EXE. Muestra un estudio completo, didáctico e interactivo de los ciclos de Rankine.

Se han creado dos versiones: una en idioma español y otra en inglés, esta última ha sido enviada a la Universidad de North Carolina (U.S.A.) para ser registrada en la biblioteca de dominio público que dicha Universidad mantiene.

- **Metodología.**

Los pasos seguidos en el desarrollo del programa pueden ordenarse de la siguiente manera:

— 1.- Creación de una base de datos que contenga información sobre los valores de las propiedades termodinámicas (presión, temperatura, densidad, entropía, entalpía, calidad de vapor) de los distintos estados posibles del agua (líquido, líquido saturado, vapor, vapor saturado y vapor supercalentado). Se trata de pasar las tablas bibliográficas de vapor a arrays de una o dos dimensiones que puedan ser accesibles por el código del programa. Varias funciones de búsqueda e interpolación conseguirán hallar los valores adecuados, y los intermedios de dos consecutivos de la tabla.

Las tablas empleadas en esta primera etapa son más pobres que las conseguidas posteriormente (y que fueron utilizadas para el modelo dinámico que en los siguientes capítulos se explicará), ya que abarcan rangos limitados para todos los estados. Así, no podemos sobrepasar temperaturas de 284 ºC en el líquido y vapor saturado, lo cual repercute en los demás estados (vapor, líquido y vapor supercalentado). Por tanto, y en cualquier caso, para nuestro primer programa RANKINE.EXE, los límites son:

$$300\ ^{\circ}C < T \text{ vapor supercalentado} < 850\ ^{\circ}C$$

$$T \text{ saturado} < 284\ ^{\circ}C$$

En posteriores modelos y programas, estos límites ya no existirán, pues habremos utilizado tablas de mucha mayor precisión.

En cuanto a la descripción e implementación de las tablas en funciones, lo explicaremos en profundidad en el próximo capítulo, ya que serán usadas en la modelación de un sistema dinámico, y además dispondrán de mejoras respecto a las aquí consideradas.

— 2.- Implementación de la formulación termodinámica de los procesos. Esto nos permitirá no sólo hallar los valores de las variables ya citadas en cada estado, sino también la cuantificación de los intercambios energéticos a lo largo del ciclo. Como resultado de este esfuerzo, obtendremos finalmente el valor del rendimiento del ciclo, que es nuestro objetivo principal.

— 3.- Estructuración de un código para ir de los menús al desarrollo de las opciones.

— 4.- Creación de las funciones y herramientas necesarias para los gráficos, cálculos, ayudas, menús, etc.

El resultado es un programa que puede describirse de la siguiente manera:

— Acceso al programa. Nos lleva a un menú de opciones principal en el cual el usuario podrá elegir el tipo de ciclo que desee estudiar, o bien abandonar el programa.

— Una vez elegido el ciclo, se pedirá que se introduzcan los valores de los parámetros de control antes mencionados. Según el ciclo que consideremos, se pedirán más o menos parámetros. Por ejemplo, para el ciclo simple, no se pedirá la temperatura del supercalentador, pues éste no está presente. Se contará con una información adicional con los rangos permitidos de los valores a introducir.

— Nos encontramos entonces con el menú específico del ciclo elegido. Contaremos con las siguientes opciones:

1.- Ayuda. Nos introduce en una explicación didáctica del proceso termodinámico que nos ocupa, mostrando los gráficos y la formulación necesaria.

2.- Datos y estadísticas. Muestra los valores de las variables termodinámicas para cada estado, las cantidades de los intercambios energéticos y, por último, el rendimiento. Tras ello, mostrará un gráfico estadístico de los balances de energía (pérdidas, energía útil, etc.).

3.- Gráfico. Nos ofrece la posibilidad de ver sobre un diagrama Presión - Entalpía, la evolución del ciclo que hemos configurado.

5.- Retorno al menú principal. Desde ahí podremos elegir otro tipo de ciclo, o bien abandonar el programa.

- **Técnica.**

  Para el desarrollo e implementación de todo nuestro estudio en un programa informático, se ha contado con las siguientes técnicas:

  — Hardware: PC-80386 ,1Mb RAM, 2 floppies de 3.5" y 5.25" y monitor color VGA. No obstante, los requisitos para la creación de RANKINE.EXE se satisfacen desde un PC XT, con 640 Kb RAM, 1 floppie 5.25 360 Kb, disco duro y monitor blanco y negro con tarjeta CGA ó Hércules.

  — Software: Turbo C v.2.0 de Borland International. Este paquete nos permitió editar y compilar nuestros códigos.

  En la figura 3.8 podemos ver las características y los códigos para la consecución de nuestra aplicación.

Programa **RANKINE.EXE**

Tamaño	Sistema Operativo	Plataforma	Lenguaje	Librería gráfica	Interactividad	Presentación
237 Kb	MS-DOS	PC compatible	C Turbo C v2.0 Borland	BGI Borland Graphic Interface	Teclado	modo TEXTO y modo GRAFICO

FICHEROS	TAMAÑO (Kb)	DESCRIPCION	COMENTARIO
RANKIS.C	15.5	Código fuente C	Contiene menús, gráficos, etc. para el ciclo simple e ideal., y funciones diversas
RANKRS.C	16.5	Código fuente C	Contiene las funciones para los ciclos real, supercalentado y recalentado.
RANKCALC.C	24.6	Código fuente C	Tablas termodinámicas y funciones para su tratamiento e interpolación
RANKGRAF.C	17	Código fuente C	Funciones para dibujar las gráficas
EXPLN1.C	30.2	Código fuente C	Contiene textos y dibujos explicativos
EXPLN2.C	7.7	Código fuente C	Contiene textos y dibujos explicativos
RANK.H	1.4	Código fuente C	Cabecera, con declaraciones de funciones.
RANKINE.PRJ	0.4	Fichero proyecto	Describe los ficheros a compilar
RANKINE.EXE	237.5	Fichero ejecutable	Aplicación obtenida

Figura 3.8

III.2.2.- GUIA DE USO DE RANKINE.EXE.-

En este apartado nos proponemos ofrecer las pautas de comportamiento del programa, su manejo y sus posibilidades.

Como requisitos "hardware", volvemos a decir que el mínimo está en un IBM PC XT compatible con 640 Kb RAM, una unidad de diskette de baja densidad y una tarjeta gráfica CGA ó Hércules, si bien se recomienda el monitor a color y tarjeta VGA.

En la figura 3.9 podemos ver esquematizado el cuerpo del programa, con sus direcciones lógicas y posibilidades de elección. A continuación, detallaremos pormenorizadamente cada una de estas características, siguiendo un orden lógico.

- Partiendo del "prompt" del DOS, ejecutamos tecleando *rankine*, lo cual nos introduce en el programa.

- Tras una sencilla pantalla de presentación, entramos en el menú principal de la aplicación (Fig. 3.10 a), en el cual se nos muestran cuatro posibles ciclos de Rankine a analizar: ciclo simple ideal, simple real, supercalentado y recalentado.

- Una vez escogida la configuración, nos introducimos en la pantalla de parámetros. Aquí hemos de dar valores a los parámetros fundamentales del ciclo, y que serán más o menos según el tipo de ciclo elegido en la anterior pantalla. Así, por ejemplo, para el ciclo recalentado, los parámetros son T_a, T_b, E_t, E_p, T_{sh} y P_h (Fig. 3.10 b), mientras que para el simple e ideal tan sólo son T_a y T_b.

- Después de dar los valores necesarios (ajustándonos a los rangos establecidos), entramos en el menú específico del ciclo considerado (figura 3.11 c). Aquí podemos pedir una explicación general, calcular las variables termodinámicas de los estados, la cuantía de los intercambios energéticos, rendimientos, gráfico de los procesos, gráficos estadísticos, etc., o bien retornar al menú principal, bien para escoger otro ciclo, bien para dar otros valores a los parámetros.

- Las pantallas de explicación vienen provistas de una o más páginas de texto (figura 3.11 d), una pantalla gráfica con el esquema de la planta (3.12 e) y un gráfico con los diagramas termodinámicos clásicos (Fig. 3.12 f).

- Si escogemos analizar el ciclo mediante la segunda opción del menú específico, nos encontramos en primer lugar con una pantalla mostrando una tabla de estados con los valores de las variables termodinámicas más representativas (Fig. 3.13 g), después las cantidades arrojadas en los balances energéticos y los rendimientos del ciclo para los casos real e ideal (figura 3.13 h), y por último, unos gráficos representando los porcentajes del rendimiento del ciclo y los balances energéticos (Fig. 3.14 i).

- Por último, la tercera opción específica del ciclo es representarlo sobre el clásico diagrama presión-entalpía, pudiendose observar la localización de los estados y los procesos termodinámicos que se producen (Fig.3.14 j).

- Para salir, desde el menú específico de ciclo pedimos volver al menú principal, desde el cual abandonaremos el programa retornando al DOS.

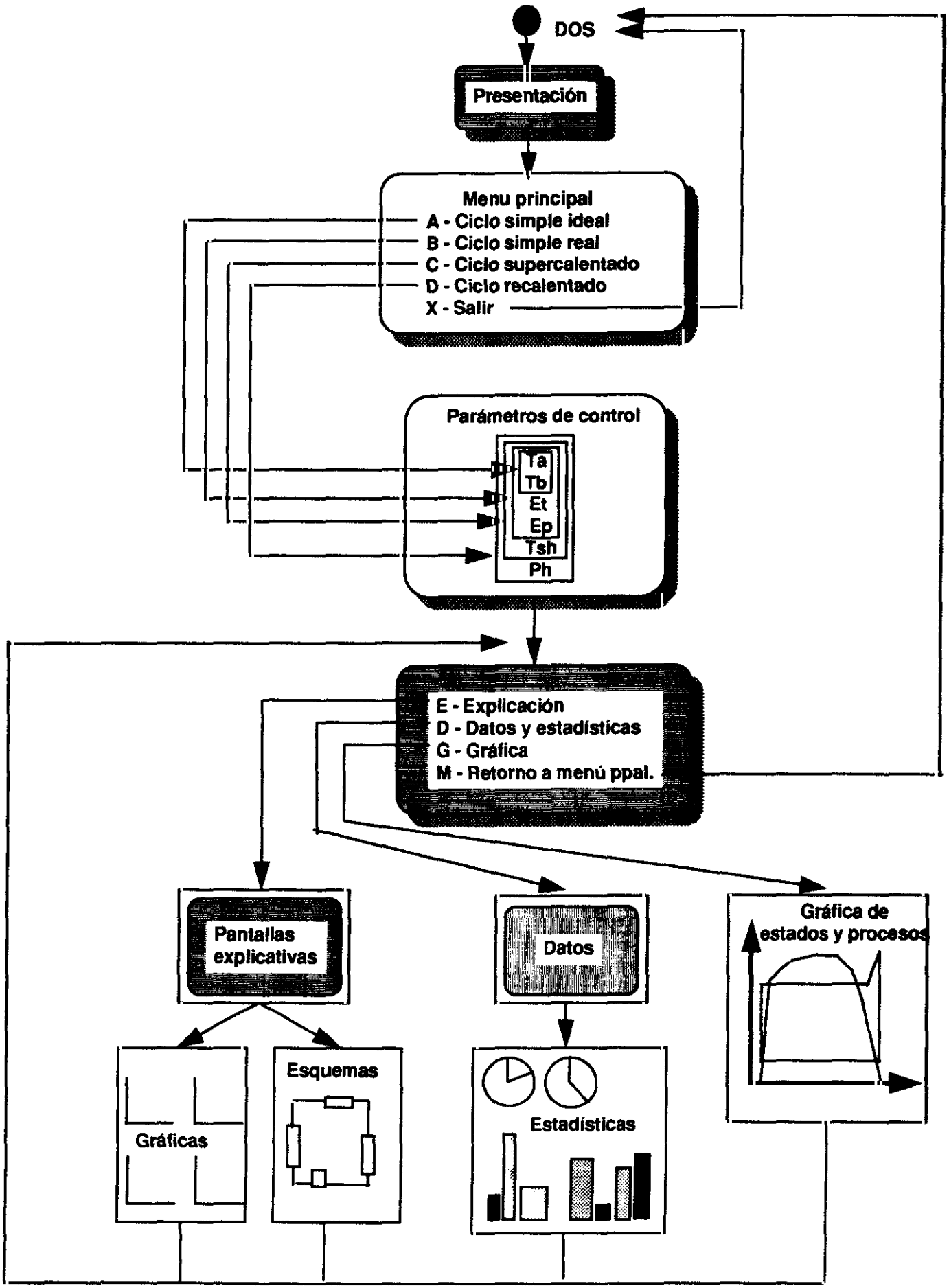

DOS
Presentación
Menu principal
A - Ciclo simple ideal
B - Ciclo simple real
C - Ciclo supercalentado
D - Ciclo recalentado
X - Salir
Parámetros de control
Ta
Tb
Et
Ep
Tsh
Ph
E - Explicación
D - Datos y estadísticas
G - Gráfica
M - Retorno a menú ppal.
Pantallas explicativas
Datos
Gráfica de estados y procesos
Gráficas
Esquemas
Estadísticas

Figura 3.9

```
                ================
**********          MENU          **********
                ================


A ===> Ideal and Simple Rankine Cycle

B ===> Real and Simple Rankine Cycle

C ===> Superheating in Rankine Cycle

D ===> Reheating in Rankine Cycle



X ===> Exit



          Type your choose option
```

a

```
                    INPUT DATAS          ((Reheated case))


Ta=> water saturation temperature into the boiler (80ºC<Ta<275ºC) (ref.:250 ºC
Tb=> condensation temperature into the condenser   (0ºC<Tb< Ta)    (ref.:20ºC
Et=> turbine efficiency       (ref.: 70 %)
Ep=> pump efficiency          (ref.: 60 %)
Tsh=>superheater temperature          (300ºC<Tsh<600ºC)           (ref.:400ºC
Ph=> reheated steam pressure        (500<Ph<5000 kPa)             (ref.:1000

                              Ta= 250

                              Tb= 20

                              Et= 70

                              Ep= 60

                              Tsh=400

                              Ph= 1000
```

b

Figura 3.10

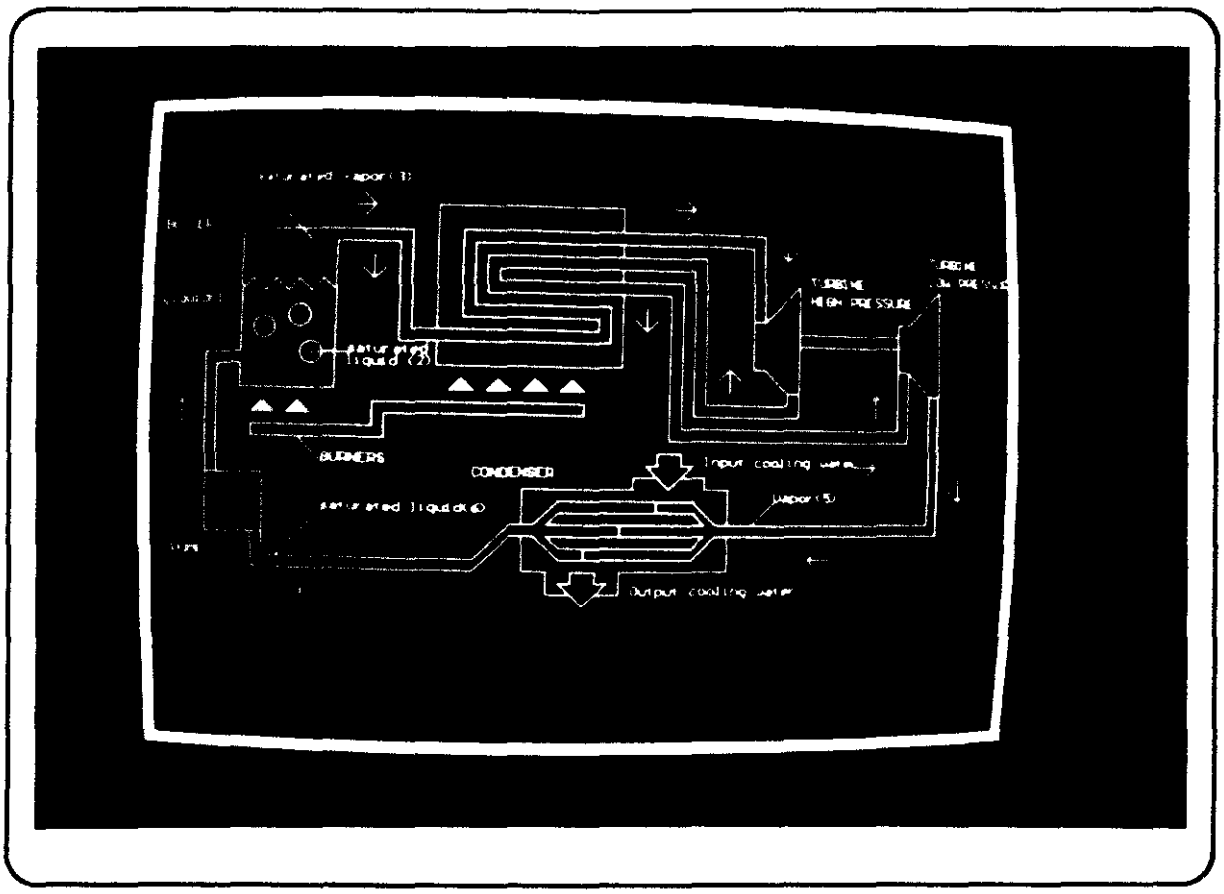
```
*************** REHEATING ***************
          ====================

E ===> Explanation

D ===> Datas and statistics

G ===> Graphic


M ===> go to Main Menu




          Type your choose option
```

c

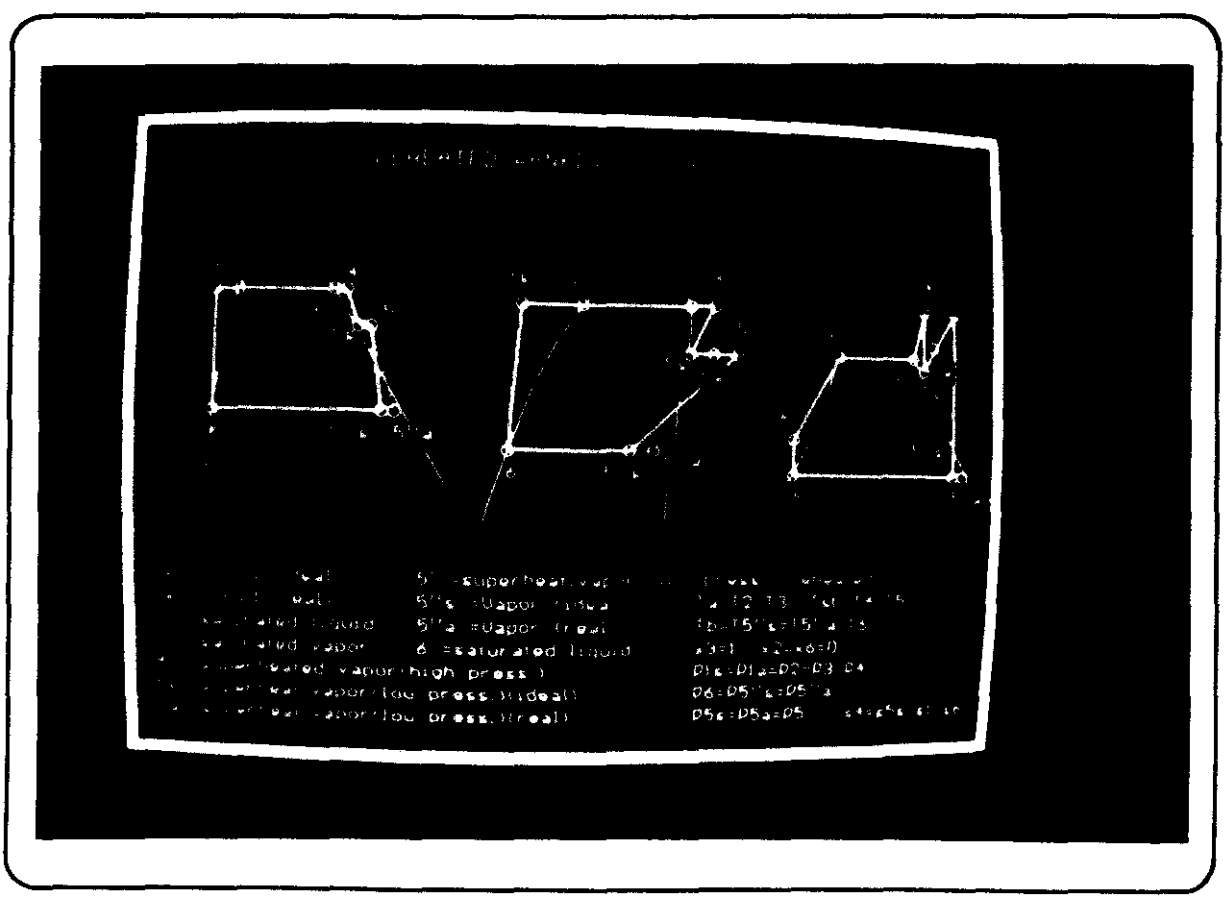
```
                        REHEATED RANKINE CYCLE
     Reheated Rankine cycle has got new characteristics. The reheated satura
vapor (4) has got a high pressure at the reheater output. Then it goes thro
a turbine ( high pressure ); at the output, we have superheated vapor at l
lpressure (5), having a cooling and reducing its temperature and pressu
according with the last turbine. After, it enters again into the superheat
reaching the Tsh temperature but keeping low its pressure, so at the superhea
output we have the reheated and with low pressure superheated vapor state (5
Then, it passes through a turbine (low pressure) at wich output we see the va
state ( 5''). The cycle continues from here in the same description of
previous cycles.
     This cycle efficiency is:

                    W(4->5)+W(5'->5'')+W(6->1)
     Ereh = - ---------------------------------
              |Q(1->2)+Q(2->3)+Q(3->4)+Q(5->5')|


     Ereh > Esup because with the two turbines we have increased the util  pc
of the cycle,and then it increases its efficiency.

*****************************************************************************
*  Options:      S ==> SCHEME          D ==> DIAGRAMS          X ==> EXIT
*****************************************************************************
```

d

Figura 3.11



e



f

Figura 3.12

STATES:
=======

	x	T (ºC)	P (kPa)	v (m3/kg)	s (kJ/(kgK))	h (kJ/kg)
1s :			3974.200		0.2965	87.8798
1a :			3974.200			90.5330
2 :	0.0	250.00	3974.200	0.001250	2.7917	1085.6000
3 :	1.0	250.00	3974.200	0.050182	6.0708	2801.1580
4 :		400.00	3974.200		6.7738	3214.5334
5s :		217.41	1000.000		6.7738	2867.5116
5a :			1000.000			2971.6181
5' :		400.00	1000.000		7.4633	3262.7000
5''s:	0.9	20.00	2.340	49.481772	7.4633	2185.4745
5''a:		20.00	2.340			2508.6422
6 :	0.0	20.00	2.340	0.001002	0.2965	83.9000

g

$$\mu rec = - \frac{W(4->5) + W(5'->5'') + W(6->1)}{|Q(1->3)+Q(3->4)+Q(5->5')|}$$

(Todas las magnitudes de trabajo y calor están en kJ/kg)

W(4->5s)=h[5s]-h[4]=-347.021864 W(4->5a)=h[5a]-h[4]=-242.915305

W(5'->5''s)=h[5''s]-h[5'])=-1077.225496 W(5'->5''a)=h[5''a]-h[5'])=-754.0578

W(6->1s)=h[1s]-h[6]=3.979804 W(6->1a)=h[1a]-h[6]=6.633006

Q(1s->2)=h[2]-h[1s]=997.720196 Q(1a->2)=h[2]-h[1a]=995.066994

Q(2->3)=h[3]-h[2]=1715.558000 Q(2->3)=h[3]-h[2]=1715.558000

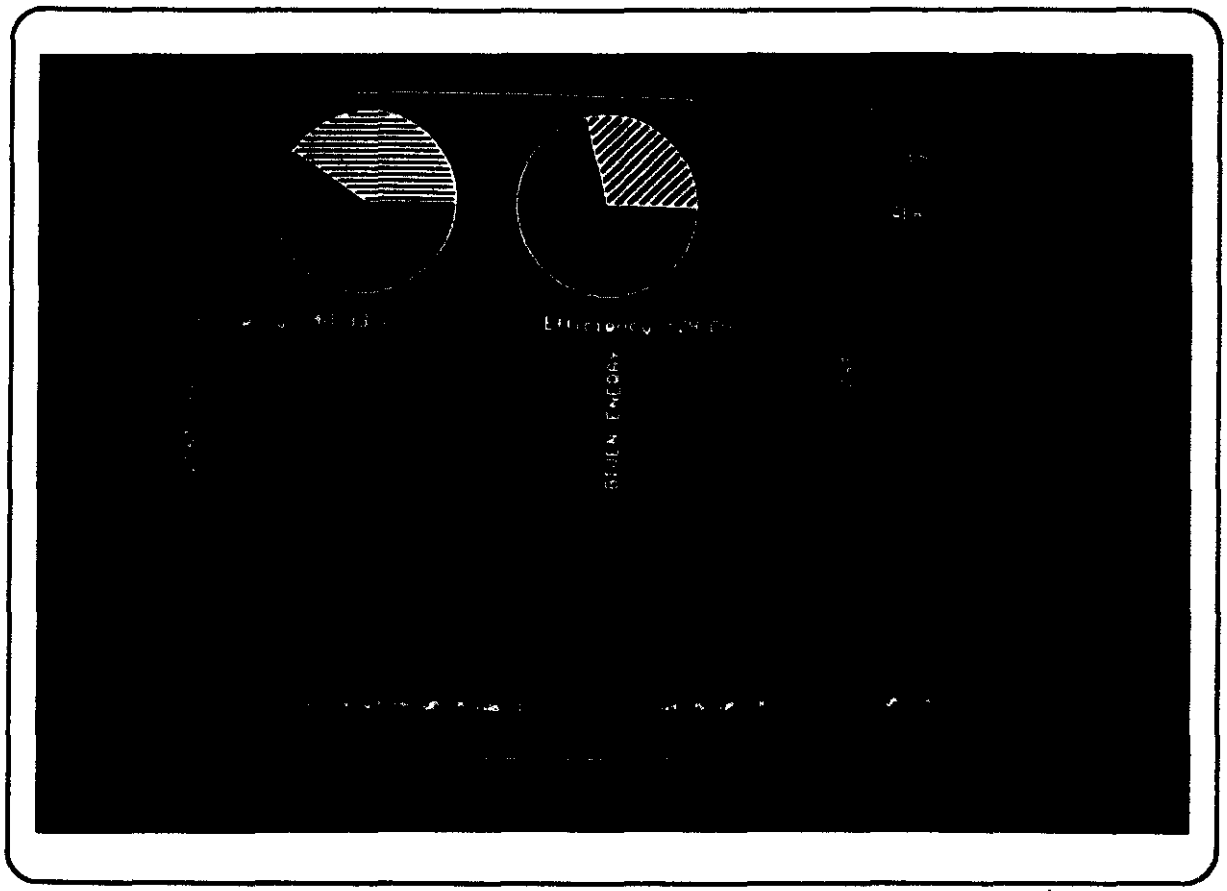
Q(3->4)=h[4]-h[3]=413.375440 Q(3->4)=h[4]-h[3]=413.375440

Q(5s->5')=h[5']-h[5s]=395.188424 Q(5a->5')=h[5']-h[5a]=291.081865

EFFICIENCY : μrec,s = 40.3274 % μrec,a = 28.9990 %

h

Figura 3.13



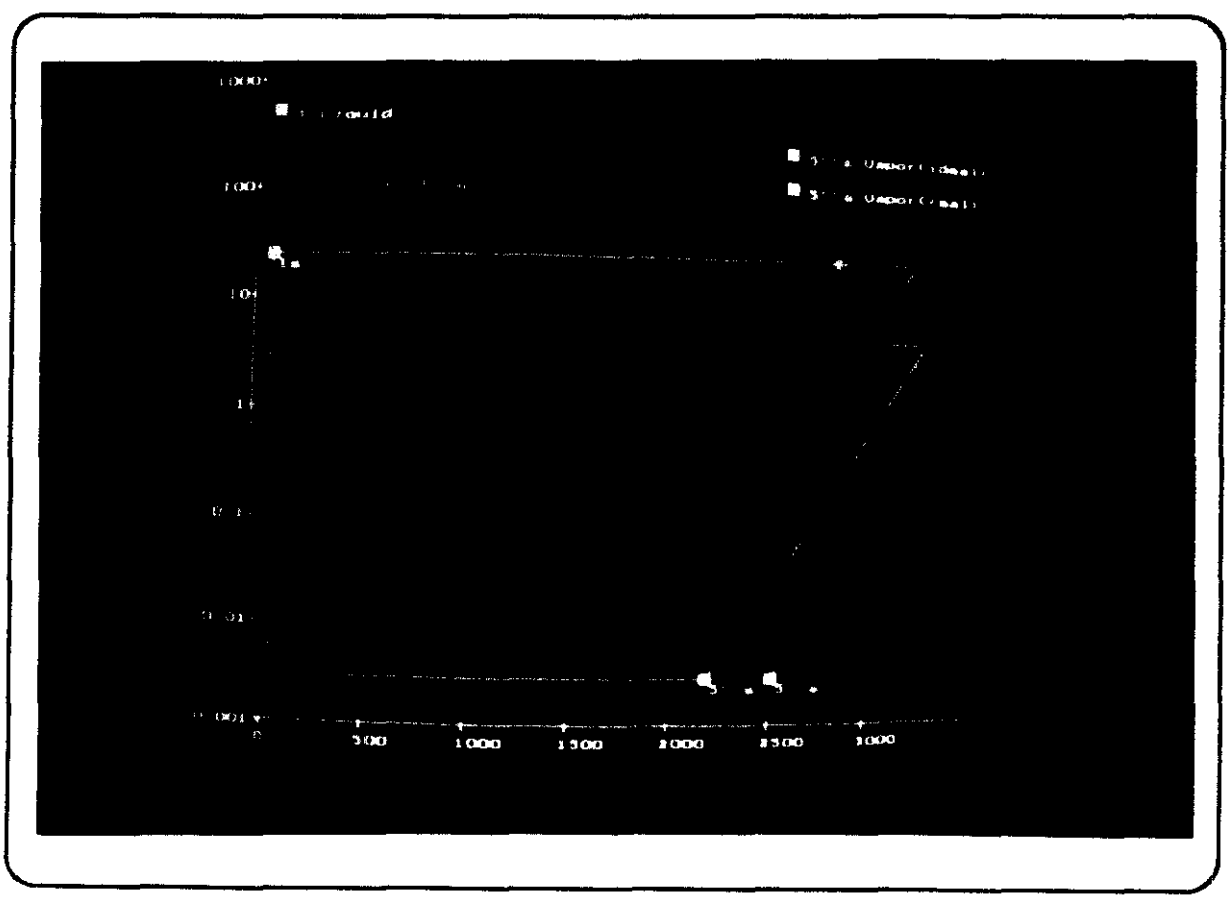


Figura 3.14

Capítulo IV:

PLANTEAMIENTO DE LA SIMULACION DINAMICA

Capítulo IV:

PLANTEAMIENTO DE LA SIMULACION DINAMICA

Tras efectuar una primera simulación, de carácter estático, que nos permitió profundizar en la teoría y ganar experiencia; emprendimos una segunda etapa, de mayor alcance, con el objetivo de realizar una simulación dinámica.

En este Capítulo, muy breve, veremos los puntos principales que definen la orientación de nuestro trabajo. Se trata de los objetivos que nos fijamos, y del planteamiento concreto de las características que queremos junto con la tecnología a incorporar. En continuidad con los primeros Capítulos, donde hicimos una discusión en el marco de la situación actual, subrayaremos algunos extremos entonces considerados.

Abrimos aquí una amplia sección de la Tesis, que expone, en los Capítulos y Apéndices que siguen, los resultados de la modelación y simulación dinámica realizadas a partir del planteamiento que ahora presentamos.

IV.1- OBJETIVOS.-

Nuestra investigación tuvo su punto de arranque observando los resultados de un grupo de investigadores que, durante años y con un considerable esfuerzo, lograron crear en el Departamento de Informática y Automática de la Facultad de Física de la Universidad Complutense de Madrid, un simulador de central térmica, visualizado en un impactante panel mímico. Consideramos que sería útil efectuar un prototipo de simulador que fuese visualizado mediante un entorno windows para estaciones de trabajo, e implementado mediante técnicas más novedosas como la programación orientada a objeto.

Por tanto, y sin tratar de competir mínimamente con el simulador ya realizado, establecimos como objetivo para nuestro trabajo, la modelación de un prototipo sencillo de planta de vapor, y su simulación, mediante programación orientada a objeto y en estaciones de trabajo con entornos windows, sirviendo así como banco de pruebas de lo que puede ofrecer y dar de sí una nueva forma de simular este tipo de sistemas.

Hemos buscado un nivel de modelación de carácter fundamental, posteriormente particularizable a plantas reales concretas. Esto nos permite un cierto grado de libertad inicial, por no estar obligados a reflejar los complicados detalles técnicos de los sistemas industriales; nos basta que nuestros resultados sean coherentes con los principios de la física termodinámica y respondan a lo que cabe esperar del comportamiento de una planta de vapor.

De esta forma, nos será posible experimentar técnicas en la modelación y simulación de este tipo de sistemas, que hasta ahora no se habían usado, dentro de nuestro ámbito de conocimiento.

Es variada la utilidad de las simulaciones, tanto para los intereses de la ingeniería, como de la didáctica académica y profesional. Por lo que a nuestra investigación respecta, nos situamos en el ámbito de los conocimientos fundamentales, que interesa, por ejemplo, al entrenamiento de operadores en sus fases conceptuales. Con nuestra simulación se contribuye a dos aspectos del aprendizaje: la adquisición de unos conocimientos básicos de los procesos termodinámicos (por ejemplo, si se aumenta el aporte calórico al líquido de la caldera mediante los quemadores, se observará que tanto la temperatura como el nivel -en ausencia de otros factores-, aumentarán), y la incorporación de hábitos de lectura de datos, seguimiento de gráficas, reconocimiento de averías, control de procesos, etc. Por supuesto, existen interesantes posibilidades de uso académico, cooperando a la formación de alto nivel mediante la necesaria componente experiencial.

IV.2.- PLANTEAMIENTO.-

Buena parte de los objetivos de nuestra investigación van dirigidos al sustrato tecnológico en la simulación de sistemas. A continuación podemos pasar revista someramente a algunas características que queremos reunir, definidas por tanto como pequeños objetivos que se suman para conseguir el resultado final, y que forman parte esencial del planteamiento de nuestro trabajo.

Modelación de elementos individuales.

Queremos que el modelo de la planta surja como la integración de modelos individuales de sub-sistemas, lo cual hará que puedan configurarse distintos sistemas globales re-usando los individuales dispuestos de otra forma. Podemos confirmar con la experiencia ganada a lo largo de esta investigación, que la programación orientada a objeto se muestra como una poderosa ayuda a este fin, muy en consonancia con la modelación que podríamos denominar "mecano".

Modelación en variables de estado.

Queremos que nuestra simulación contemple la mayor cantidad posible de variables termodinámicas temporales. Si bien visualizaremos la evolución de algunas seleccionadas, el cálculo de las demás nos asegura su conocimiento en caso de estar interesados en su comportamiento concreto. En nuestro modelo podremos considerar varios conjuntos de variables: de control, de determinación inicial, de valores fijos, de visualización, de cálculo, etc.

Modelación dinámica.

Además del estado estacionario, deseamos saber cómo son los comportamientos de carácter transitorio, como por ejemplo, los que llevan a un estado final estable a partir de cierto estado inicial predeterminado por nosotros. La dinámica del proceso es nuestro motivo de estudio fundamental.

Utilización informática de las tablas de vapor.

Dada la complejidad del comportamiento del agua, se hace imprescindible utilizar las tablas termodinámicas como fuente de información precisa. Su manejo suele ser tedioso y lento, pues aparte de la búsqueda de un dato en las tablas, casi siempre es necesario hacer las interpolaciones adecuadas. Nos propusimos, por una parte, la elaboración de una base de datos que implementase en *arrays* de una o dos dimensiones dichas tablas y, por otra parte, la elaboración de funciones que gestionasen esta base de datos para tanto la búsqueda como la interpolación de los valores óptimos.

Programación Orientada a Objeto.

Queremos utilizar esta nueva técnica de programación como plataforma de pruebas para futuras investigaciones. Básicamente, lo que queremos saber es qué grado de naturalidad se consigue en la implementación de los modelos mediante objetos.

La POO, al introducir conceptos como objetos y mensajes, nos lleva a una nueva filosofía de programación que se ajusta a una forma más natural de tratar ciertos problemas. Con ella podemos conseguir re-usabilidad, multi-funcionalidad, etc. Un objeto puede crearse aisladamente para trabajar con él, pero también puede asociarse con otros objetos para que trabajen en equipo, o para que formen parte de un objeto mayor que los englobe, o para que sean dirigidos y controlados por otro objeto, o para que pueda ser reutilizado en otras tareas futuras, etc. Se abre la posibilidad de crear una biblioteca de clases para su uso en distintas aplicaciones.

En la modelación clásica se parte de un cierto sistema que realiza unas tareas concretas con un determinado número de elementos. La modelación se efectúa entonces mediante programas construidos para atacar este específico sistema. ¿Qué ocurre entonces si el sistema tiene otras configuraciones o, si modificando su disposición tenemos otros sistemas con otros comportamientos? Pues que entonces, hemos de hacer otros programas, quizá parecidos o modificados del anterior, pero distintos en cuanto a sus elementos. Cuanto más distintos sean dos sistemas, los programas para modelarlos también, con su consiguiente complejidad en cuanto a su comprensión, lo cual obliga a un estudio más detallado del código para saber qué hace y por qué.

En cambio, la POO establece unas ventajas específicas en el campo de la modelación, que permitirá modelar sistemas distintos con programas parecidos; en la biblioteca de clases encontraremos la clase adecuada para nuestro requisito concreto, haciendo que el código sea más elegante y modular. Mediante herencia, pueden especializarse y enriquecerse clases existentes para adaptarlas a casos particulares. El esfuerzo a la hora de utilizar POO es, como su propio nombre indica, orientar el programa a una visión en base a objetos y las relaciones entre ellos.

Coordinador (Scheduler).

Cuando se agrupan modelos de sub-sistemas, para componer el modelo de una planta, aparece el problema de hacer interactuar los componentes de una forma adecuada. No debe olvidarse que el ordenador clásico sólo tiene una central de proceso, y que los sucesos que ocurren al mismo tiempo en la naturaleza, tienen que ser simulados a partir de un mecanismo secuencial. De las diversas alternativas para llevar a cabo la interacción, nos decidimos por un agente central: un coordinador o gestionador que conoce cómo es la planta y que alcanza todos los mecanismos del programa, para controlar las funciones esenciales de la simulación, como es el contar el tiempo, controlar los valores, establecer las relaciones tanto internas como externas, etc. La ventaja de este principio es que tenemos todo lo esencial centralizado de forma que no es necesario perderse por caminos que se van ramificando cada vez más, sino que basta con saber qué es lo que hace y controla este gestionador del programa. El coordinador (Scheduler) será programado, asímismo, como un objeto.

X-Window.

Queremos no sólo que el aspecto gráfico de nuestra aplicación sea atractivo y manejable, sino que responda a unos estandards que la hagan compatible con un amplio rango de sistemas y entornos tanto de ejecución como de comunicación. Esta es, pues, la razón de utilizar X-Windows en lugar de otras plataformas gráficas. Por otra parte, X-Windows nos proporciona, además de todo lo gráfico, la posibilidad de incorporar a nuestro programa una serie de librerías para el manejo de ratón y gestión de ventanas, etc.

Estaciones de trabajo.

Entendidas como la principal herramienta informática profesional, y presente en los ambientes industriales, las estaciones de trabajo permiten un alto grado de difusión de las aplicaciones (mediante el standard X-Window) y una rapidez de cálculo, indispensable en una simulación que, como es nuestro caso, descansa en la gestión de una base de datos (tablas termodinámicas) y en una abundante presentación gráfica de resultados durante el tiempo de la simulación. Son normalmente superiores a los PC compatibles. De ello contamos con la experiencia de tener una misma aplicación para PC's y para *workstations*, lo que nos ha permitido calibrar las posibilidades de ambos sistemas de proceso.

Control.

Vamos a considerar dos tipos de control. En primer lugar, y como el más importante, como objetivo fundamental, queremos un control interactivo por parte del usuario, sobre una serie de parámetros de control mediante los cuales se alteren los procesos, creándose unos transitorios para el análisis de incidencias, averías, etc. Este control interactivo será efectuado a través del teclado y del ratón. La interactividad será hecha en tiempo de simulación, y abarcará un suficiente número de acciones (aumentar o disminuir caudales, aportes energéticos, etc.) y, como ya hemos dicho, será de gran utilidad tanto en la simulación de eventos como en tratar de controlar una simulación para conseguir unas evoluciones deseadas. Podríamos considerar también como un cierto control interactivo a las capacidades de parar y reanudar la simulación, abortarla, configurar su estado inicial, almacenar datos, buscar ficheros, abrir y cerrar ventanas, etc.

En segundo lugar, y menos importante, un control automático de tipo software que asigne valores determinados a los dispositivos cuando se alcanzan unas condiciones también determinadas; esto será necesario para evitar que, por ejemplo, algunas variables lleguen a salirse de los rangos establecidos por las tablas de vapor, lo cual introduciría errores de difícil interpretación. Este control automático ha de ser, por otra parte, coherente con la realidad. Así, por ejemplo, para evitar que la temperatura del vapor supercalentado aumente hasta sobrepasar los límites conocidos, cerramos inmediatamente las espitas del quemador que suministra el aporte energético al supercalentador.

Gráficos.

Necesitamos que nuestra aplicación disponga de la mayor visualización posible, rehuyendo los listados de números, los avisos, etc. Queremos que se monitorizen las evoluciones de las variables más importantes, tales como los niveles de líquido en la caldera y condensador, las temperaturas de los vapores, el rendimiento del ciclo de vapor, etc. También queremos dotar de apariencia gráfica los menús de opciones, para dar valores a los parámetros, para abrir ficheros, etc., y también ofrecer ayudas, esquemas, etc. de una forma más atractiva que si fuese tan sólo texto. Además exigimos que estos gráficos tengan un suficiente nivel de interactividad tanto con el teclado como con el ratón.

Almacenamiento y recuperación de simulaciones.

También nos propusimos como requisito el poder guardar en ficheros simulaciones enteras, con el objeto de poder recuperarlas de nuevo. Una de las muchas ventajas de hacer ésto es que, la primera vez que se simula, el tiempo invertido en cálculo es considerable, pero después, importando los valores escritos en los ficheros, podremos ver la evolución de la simulación a una velocidad muy superior, ya que tan sólo necesitamos leer de laos ficheros.

Conviene aquí mencionar los cuatro tiempos que se invierten en el proceso de la simulación:

— Tiempo de cálculo.
  Es el que se gasta en la gestión de la base de datos, en la interpolación de valores y en el cálculo de ecuaciones. Este tiempo será mayor o menor dependiendo de la velocidad de procesamiento del microprocesador, y de si el tratamiento de las operaciones aritmético-lógicas está ayudado por un co-procesador matemático.

— Tiempo de almacenamiento.
  Es el invertido en guardar primero un valor en el buffer para a continuación escribirlo en la cola de un fichero. Implica el tener que esperar a que el dato se haya escrito en el soporte magnético u óptico, lo cual es una acción "mecánica".

— Tiempo de lectura.
  Es el empleado en leer un valor de una posición de un fichero, para asignarlo a la variable del programa. Este tiempo es inferior al de almacenamiento, pues no implica acciones mecánicas.

— Tiempo de representación.
  Es el necesario para que el programa ponga sobre la pantalla los "pixels" adecuados. Dependerá tanto de la memoria RAM como de la tarjeta gráfica.

Podemos considerar, así, tres acciones en la simulación del sistema, que invierten tiempos distintos. De mayor a menor tiempo empleado, tenemos:

— Simulación con almacenaje.

Es el proceso de efectuar la simulación junto con el de, cada cierto tiempo, tomar el dato y escribirlo sobre el fichero. Por tanto, sumamos tiempos de cálculo, de almacenamiento y de representación.

— Simulación.

Es la simulación típica. Antes de guardar una simulación en un fichero, conviene hacerla "a solas" con el objeto de ver su comportamiento por si fuera "digna" de guardarse. El tiempo empleado es para cálculo y representación.

— Lectura de la simulación.

Elegida una simulación de una lista de ficheros ya existentes, ésta aparecerá como si estuviese realizándose por primera vez, pero a un ritmo muy superior. El tiempo empleado es de lectura y representación.

IV.3.- METAS.-

A continuación fijaremos los requisitos de nuestras aplicaciones. Básicamente, ambas responderán a los mismos objetivos, si bien algunos planteamientos serán distintos, por una razón que justificaremos.

Las aplicaciones que vamos a concretar como metas serán los programas informáticos

PLANTA.EXE y **planta**.

que tendrán unos puntos en común, y otros elementos diferenciadores, y que podemos ver esquematizados en la tabla 4.1.

La motivación de realizar simulaciones del mismo modelo bajo plataformas y entornos distintos es la de poder ofrecer un mayor rango de aplicabilidad al potencial usuario, tanto a nivel de PC como de estaciones UNIX, tanto para un interés académico en un ambiente universitario como para un entorno de aprendizaje industrial. Mediante el programa PLANTA.EXE podemos conseguir mayor grado de difusión a nivel PC, mientras que con *planta* estamos capacitados para ofrecerlo sobre un amplio rango de máquinas que trabajen con distintos sistemas operativos UNIX (a condición de tener instaladas las librerías X-Windows y GNU g++ necesarias).

PROGRAMA	**PLANTA.EXE**	**planta**
PLATAFORMA	PC compatible	Sun Sparc 1+
SISTEMA OPERATIVO	MS-DOS 3.3	UNIX (SunOs 4.1)
TAMAÑO (Kb)		
PROGRAMACION ORIEN-TADA A OBJETO	Sí	Sí
LENGUAJE	Borland Turbo C++ v.2.0	GNU G++ v.1.39
GRAFICOS	Borland BGI	X-WINDOW v11 R4
INTERACTIVIDAD EN TIEMPO DE SIMULACION	Teclado	Teclado y Ratón
SCHEDULER	Sí	Sí
ALMACENAMIENTO Y RECUPERACION DE LAS SIMULACIONES	Sí	Sí

Tabla 4.1

Capítulo V:

MODELACION DINAMICA

Capítulo V:

MODELACION DINAMICA

Expondremos en este Capítulo una aportación crucial de nuestra investigación: la modelación dinámica mediante programación orientada a objeto, de los sub-sistemas que intervienen en la implantación y ejecución del ciclo de Rankine.

Se trata de una pieza fundamental para la simulación que hemos creado subsecuentemente.

Dividimos la exposición en tres apartados.

El primero se dedica brevemente a custiones previas de nomenclatura e hipótesis.

En el segundo se encuentra el núcleo de la modelación efectuada. Presentaremos con gran detalle la modelación de cada sub-sistema: caldera, turbina, condensador, etc.

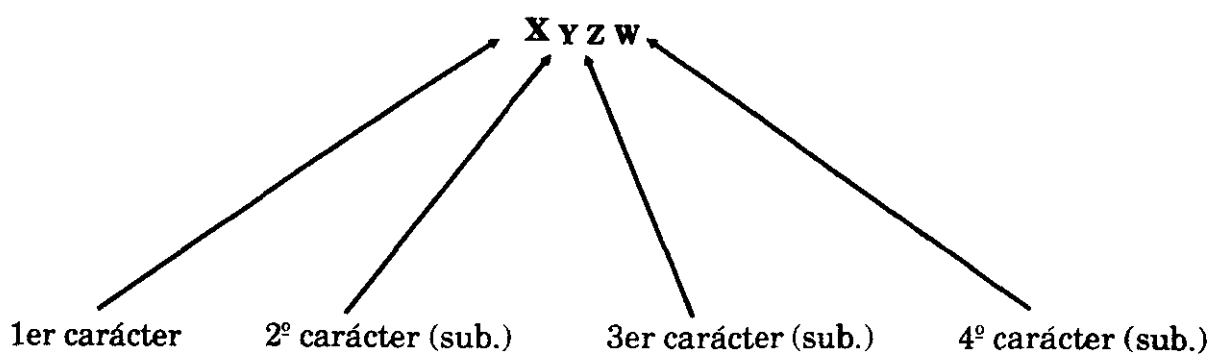
Por último, el tercer apartado se centra en la gestión de las tablas termodinámicas. Presentamos cómo hemos organizado los datos, y las rutinas que hemos creado para su tratamiento.

V.1.- CONSIDERACIONES PREVIAS.-

En este apartado sentaremos las bases notacionales y las acotaciones teóricas necesarias para la descripción de nuestros modelos. En primer término, estableceremos la nomenclatura de nuestras variables, y el sistema de unidades que emplearemos de ahora en adelante. A continuación, revisaremos algunos supuestos teóricos que afectan a la física de nuestros sistemas.

V.1.1.- NOMENCLATURA Y UNIDADES.-

Para normalizar en lo posible nuestro trabajo, y hacerlo más fácil de entender, consideraremos que un parámetro o variable cualquiera está descrito mediante una serie de caracteres (subíndices a partir del segundo).



- El primer carácter se refiere a la magnitud física considerada. A continuación damos la lista de las que usamos, junto con sus unidades:

Carácter	Unidad	Descripción
W	Kg/s	flujo másico (caudal)
t	s	tiempo
d ó ρ	Kg/m^3	densidad
m	Kg	masa
V	m^3	volumen
P	KPa	presión
T	°C	temperatura
h	KJ/Kg	entalpía
s	KJ/(Kg.°K)	entropía
Q	KJ/s	flujo de energía
R	m	radio
L	m	longitud
n	m	nivel

- El segundo carácter hace referencia al estado de la sustancia a la cual pertenece la magnitud:

Carácter	Descripción
v	vapor
l	líquido
f	fuel (combustible)

- El tercer carácter (si se considera) informa de si la variable se refiere a la entrada o a la salida de un sistema concreto:

Carácter	Descripción
e	entrada
s	salida

- El cuarto carácter (que sería el tercero si no se hace referencia a entrada o salida), y siguientes, toma en consideración la ubicación o pertenencia de la variable en un lugar o dispositivo:

Carácter	Descripción
c	caldera
a	alimentación
t	turbina
cn	condensador
s	supercalentador
b	bomba

V.1.2.- SUPUESTOS.-

Vamos a suponer una serie de acotaciones físicas que nos permitirán elaborar el modelo; estas limitaciones lo alejan de la realidad sólo hasta cierto punto, pues con todo el modelo conserva lo esencial del comportamiento que cabe esperar del sistema real.

- La caldera se halla, en su instante de partida, en condición de equilibrio saturado, es decir, el vapor que se produce está saturado, y existe parte de líquido saturado. Supondremos que este equilibrio se mantiene durante la simulación.

- No vamos a considerar los modelos de las tuberías en cuanto a pérdidas de calor y otros fenómenos relacionados con el transporte. Así, consideraremos que el estado que abandona un dispositivo es el mismo que entra en el siguiente, asumiendo de esta manera unos conductos o tuberías ideales.

- No consideramos los efectos de las paredes de los recintos en cuanto a pérdidas de calor por conducción, radiación, etc., ni en cuanto a cualquier otro fenómeno que a ellas se puedan deber. Nos atenemos, de esta forma, sólo al fluído de trabajo.

- Los aportes caloricos ofrecidos por los quemadores son, en principio, constantes, pudiendo variar a voluntad del operario a lo largo del tiempo, debido a que las ecuaciones van a ser resueltas para cada instante de la simulación, con lo cual podemos en un instante determinado cambiar el valor de estas magnitudes. Otro tanto decimos de los caudales regulados por las válvulas. Vamos a suponer que las válvulas son unos dispositivos que sólo dejan pasar una determinada cantidad de masa por segundo (tanto para líquidos como para vapores), y que estos valores pueden ser fijados o variados manualmente (en la simulación) a lo largo del tiempo (imaginando una apertura o cierre de estas válvulas); de esta forma, fijamos la cantidad de masa que abandona por segundo un dispositivo y que es la misma que entra en el siguiente, con lo cual, al usar estas peculiares válvulas, simplificaremos los cálculos matemáticos.

- El supercalentador no afecta al caudal de vapor, entendiendo que este caudal es el mismo a la entrada y a la salida; también supondremos, por tanto, que la cantidad de masa de vapor que en todo instante permanece en el interior, es constante.

V.2.- MODELOS INDIVIDUALES.-

Vamos a establecer los modelos individuales requeridos para la construcción de la planta. No sólo se trata de dispositivos mecánicos tales como una turbina o un quemador, sino también de objetos constructivos como el agua en sus distintos estados.

Los modelos teóricos partirán de un estudio que descansa sobre los principios de la termodinámica. A partir de éstos, se crearán modelos orientados a objeto, implementados en clases en el lenguaje C++.

V.2.1.- VAPOR.-

MODELO TEORICO.

Vamos a modelar el vapor a través de una serie de funciones y variables que corresponden a sus propiedades termodinámicas y definen sus estados. Para mayor información de estas funciones (que gestionan la base de datos con las tablas de vapor, e interpolan valores), remitimos al apartado V.3.

El vapor puede tener tres estados: normal (o, simplemente, vapor), saturado y supercalentado.

MODELO ORIENTADO A OBJETO.

En la figura 5.1 podemos ver esquematizada la clase *Vapor*, que contará con los siguientes miembros:

— *Protegidos:*

presión, temperatura, densidad, entalpía, entropía, calidad y volumen específico como variables que definen sus características termodinámicas. Una variable especificará su estado (0 como vapor, 1 vapor saturado y 2 supercalentado). Una cadena de caracteres etiquetará el nombre del estado en que se halla el vapor.

— *Públicos:*
una serie de funciones buscarán en las tablas de vapor ciertas propiedades (setPyh, setPys, ver apartado V.3), y otras devolverán los valores de éstas (dar_T, dar_P, dar_v, dar_s, dar_h, dar_x, dar_d y dar_estado).

V.2.2.- LIQUIDO SATURADO.-

MODELO TEORICO.

En el apartado V.3 se encontrarán las funciones que gestionan las tablas.

MODELO ORIENTADO A OBJETO.

En la figura 5.2 puede verse esquematizada la clase *Líquido_saturado.*

— *Protegidos:*
variables densidad, temperatura, entalpía, entropía, presión, masa, Volumen y volumen_específico.

— *Públicos:*
setPyV (establece el valor de todas las propiedades termodinámicas cuando se conocen la presión y el volumen), y otras funciones diversas.

V.2.3.- VAPOR SATURADO.-

MODELO TEORICO.

Decimos lo mismo que en los dos apartados anteriores.

MODELO ORIENTADO A OBJETO.

Ver esquema de la clase *Vapor_saturado* en la figura 5.3:

— *Protegidos:*
variables densidad, temperatura, entalpía, entropía, presión, masa, Volumen y volumen_específico.

— *Públicos:*
setPyV establece el valor de las propiedades termodinámicas; otras funciones asignan esos valores.

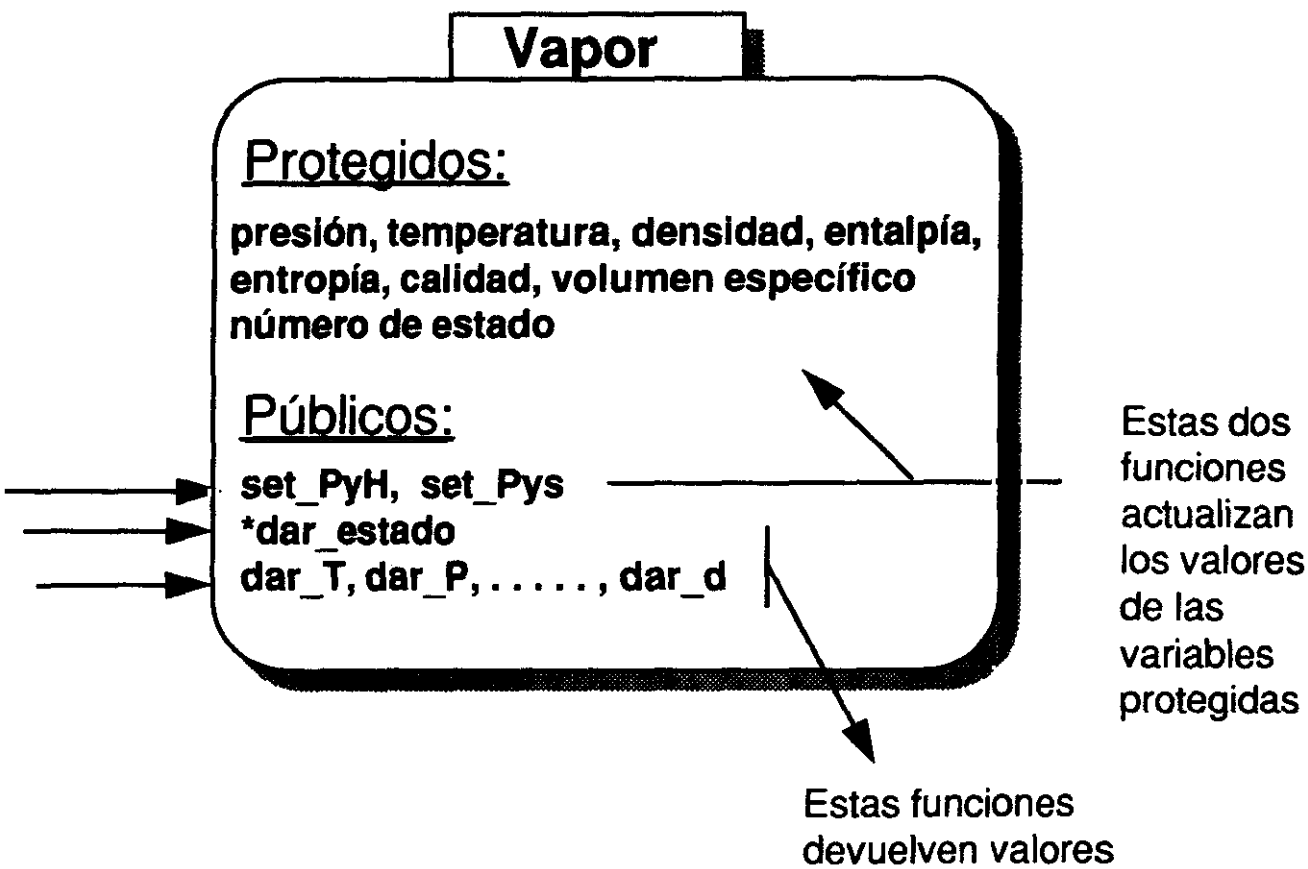

Vapor
Protegidos:
presión, temperatura, densidad, entalpía, entropía, calidad, volumen específico número de estado
Públicos:
set_PyH, set_Pys
*dar_estado
dar_T, dar_P, ... , dar_d
Estas dos funciones actualizan los valores de las variables protegidas
Estas funciones devuelven valores

Figura 5.1

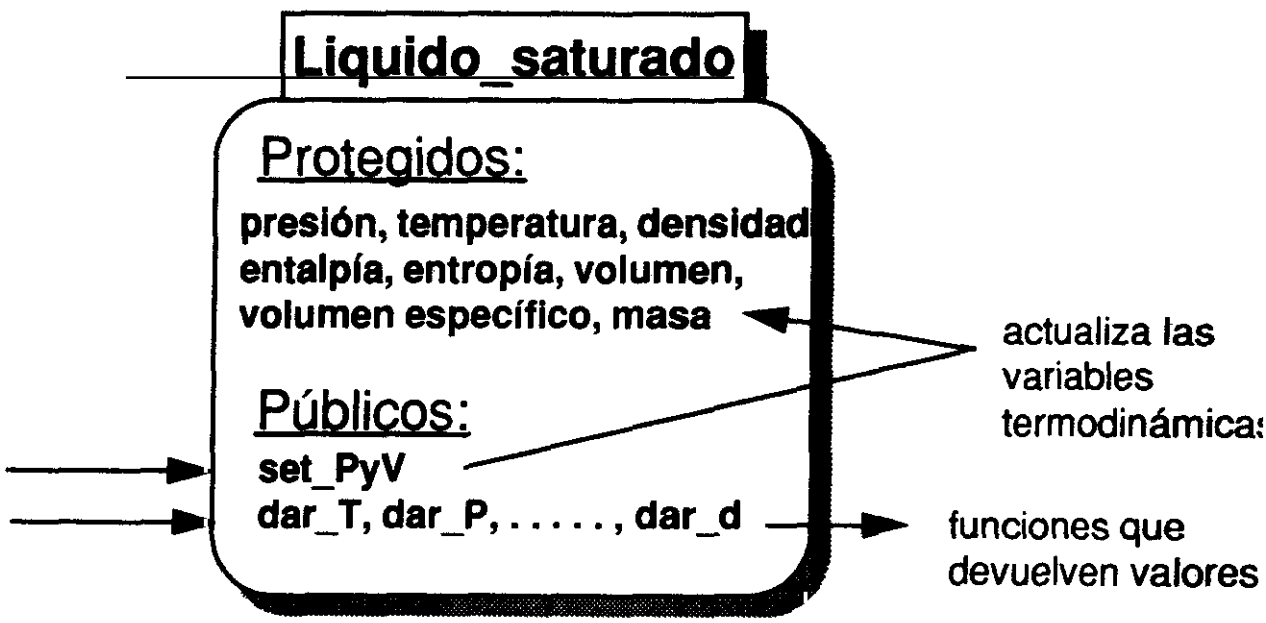

Liquido_saturado
Protegidos:
presión, temperatura, densidad
entalpía, entropía, volumen,
volumen específico, masa
Públicos:
set_PyV
dar_T, dar_P, ... , dar_d
actualiza las
variables
termodinámica
funciones que
devuelven valores

Figura 5.2

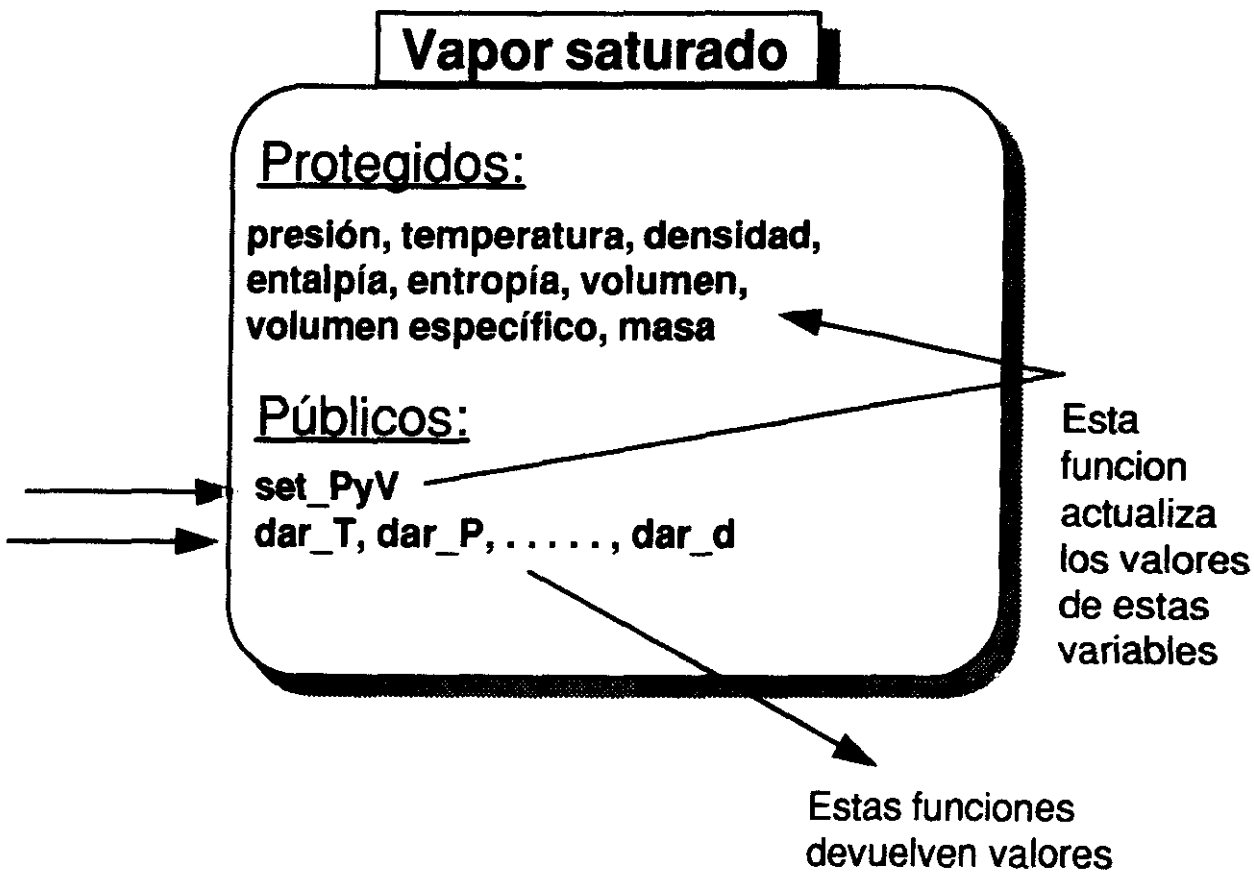

Vapor saturado
Protegidos:
presión, temperatura, densidad,
entalpía, entropía, volumen,
volumen específico, masa
Públicos:
set_PyV
dar_T, dar_P, ... , dar_d
Esta
funcion
actualiza
los valores
de estas
variables
Estas funciones
devuelven valores

Figura 5.3

V.2.4.- QUEMADOR.-

MODELO TEORICO.

Vamos a suponer al quemador como un dispositivo que aporta Q calorías por segundo. El valor de Q puede ser regulado (abstractamente, abriendo o cerrando unas espitas de gas si la planta es de combustión de gas natural, o aumentando o disminuyendo el suministro de carbón al hogar, si la planta es de carbón, etc.).

Q_{ec} será el aporte a la caldera, y Q_{es} al supercalentador.

MODELO ORIENTADO A OBJETO.

En la figura 5.4 está representada la clase *Quemador*, con los siguientes miembros:

— *Protegidos:*
Q (flujo de calorías).

— *Públicos:*
funciones para dar y variar el valor de este aporte energético.

V.2.5.- VALVULA.-

MODELO TEORICO.

Consideraremos a una válvula tan sólo como un dispositivo que nos regula el caudal de fluído (líquido o vapor) que pasa a través de un conducto. Así, si en medio de la tubería que conecta los dispositivos A y B hay una válvula regulada al valor W, diremos que el flujo de fluído saliendo de A y entrando en B es el mismo, y de valor W.

MODELO ORIENTADO A OBJETO.

En la figura 5.5 podemos ver representada la clase *Válvula.*

— *Protegidos:*
W (caudal de fluído).

— *Públicos:*
funciones para dar y variar el valor de este caudal.

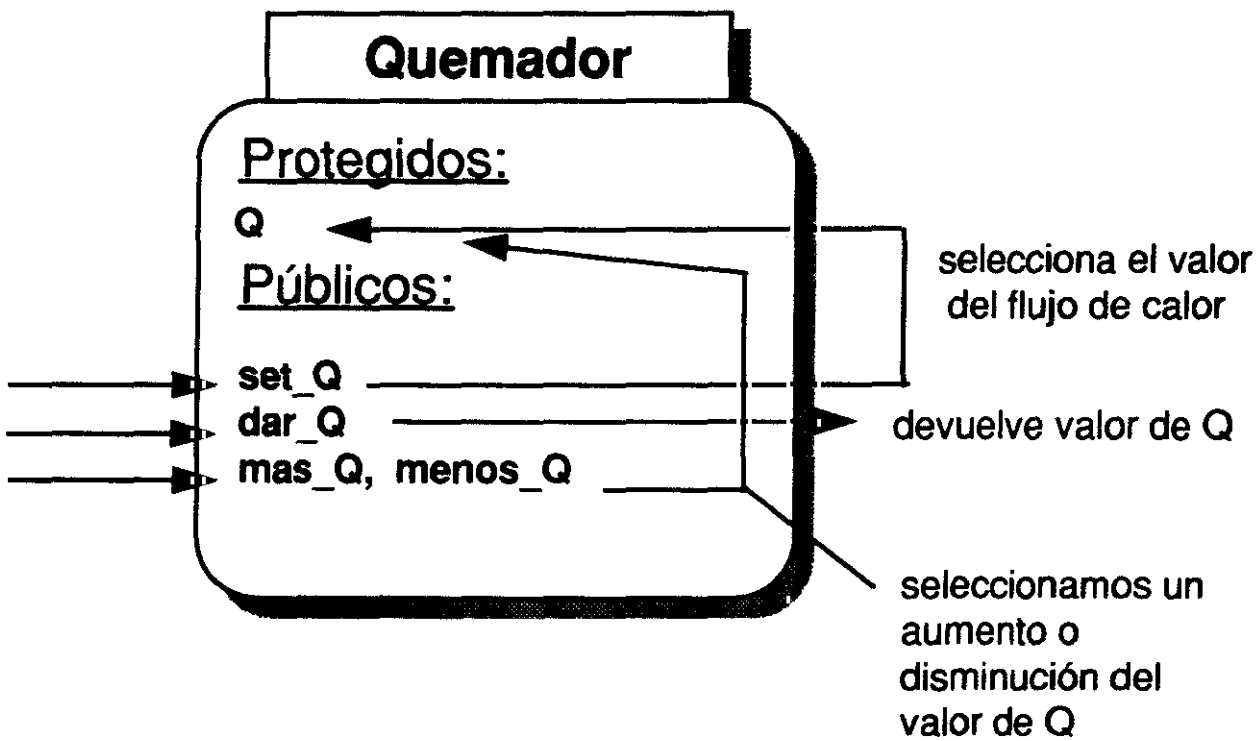

Quemador
Protegidos:
Q
Públicos:
set_Q
dar_Q
mas_Q, menos_Q
selecciona el valor del flujo de calor
devuelve valor de Q
seleccionamos un aumento o disminución del valor de Q

Figura 5.4

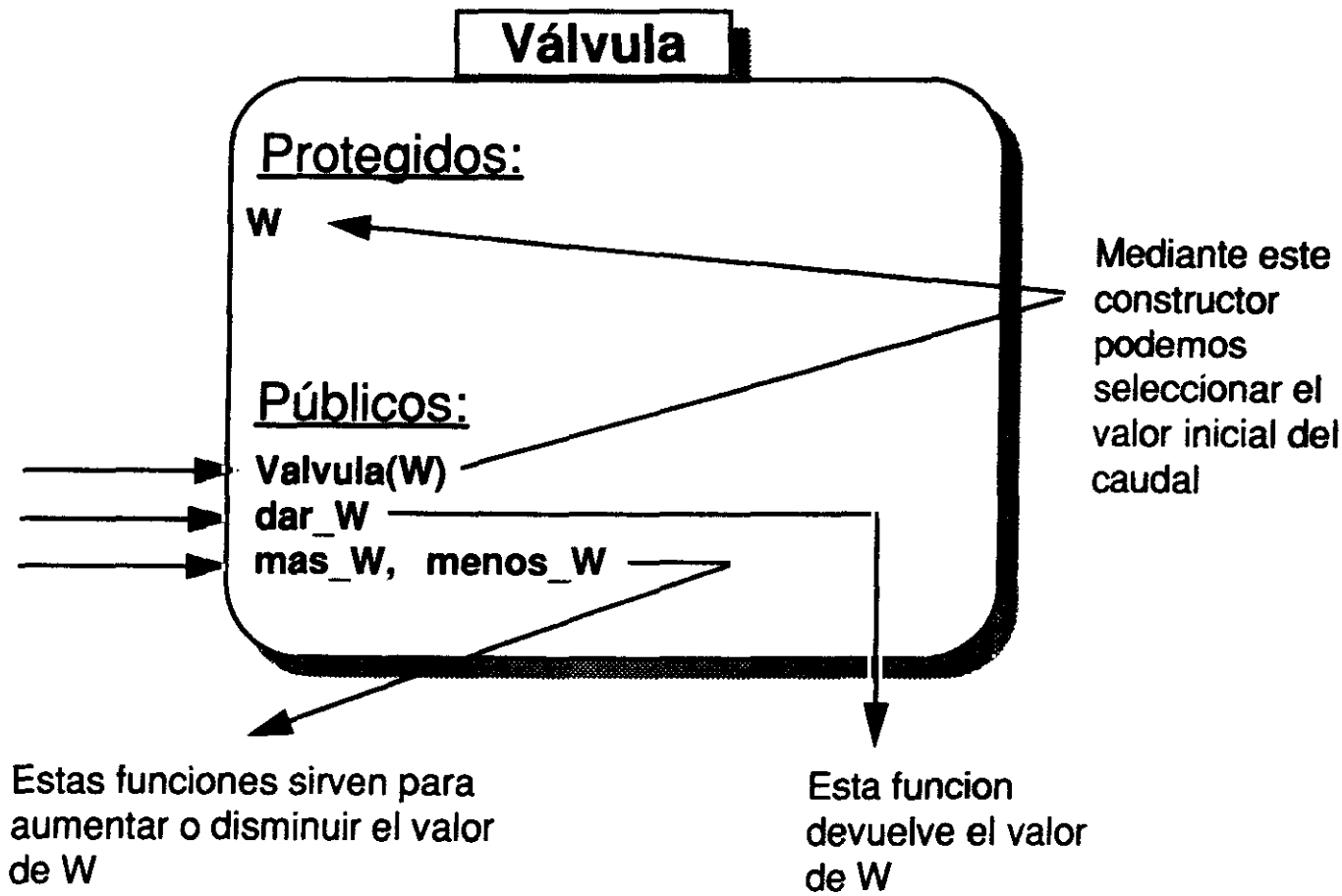

Válvula
Protegidos:
W
Públicos:
Valvula(W)
dar_W
mas_W, menos_W
Mediante este constructor podemos seleccionar el valor inicial del caudal
Estas funciones sirven para aumentar o disminuir el valor de W
Esta funcion devuelve el valor de W

Figura 5.5

V.2.6.- FUENTE DE LIQUIDO.-

MODELO TEORICO.

Usaremos este dispositivo como un ente que nos suministra agua en estado líquido a cualquier presión, pero con un valor determinado de entalpía.

MODELO ORIENTADO A OBJETO.

En la figura 5.6 está representada la clase *Fuente_de_líquido.*

— *Protegidos:*
h (entalpía).

— *Públicos:*
funciones para dar y variar el valor de esta entalpía.

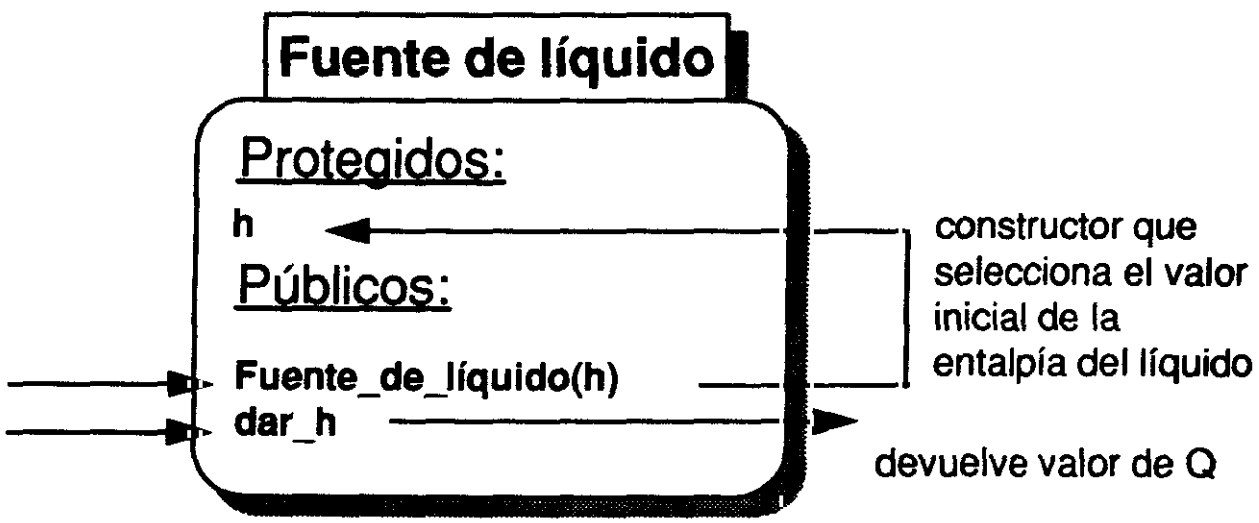

Fuente de líquido
Protegidos:
h
Públicos:
Fuente_de_líquido(h)
dar_h
constructor que selecciona el valor inicial de la entalpía del líquido
devuelve valor de Q

Figura 5.6

V.2.7.- CALDERA.-

MODELO TEORICO.

En la figura 5.7 podemos ver el esquema de nuestra caldera. Consiste en un recinto de forma cilíndrica, de altura L_c y radio R_c. En su interior tenemos agua en los estados líquido, líquido saturado y vapor saturado. Por una parte, recibe un aporte de agua en estado líquido (caudal W_{lec} con una entalpía h_{lec}), y por otro escapa vapor saturado (caudal W_{vsc} con una entalpía h_{vsc} ó h_{vc}). Además, está recibiendo un flujo de energía de Q_f calorías por segundo provenientes de un quemador u hogar.

Consideramos que, en todo momento, estamos en equilibrio saturado en el interior de la caldera, lo cual significa que el agua en estado líquido que entra en la caldera pasa a ser saturado al quedar dentro (es calentado por Q_f hasta su temperatura de saturación).

Balance de masa.

La variación en el tiempo de la masa total de fluído es igual al caudal de entrada menos el de salida, luego:

$$\frac{d(m_{vc} + m_{lc})}{dt} = W_{lec} - W_{vsc}$$

o bien

$$\rho_{vc}\frac{dV_{vc}}{dt} + V_{vc}\frac{d\rho_{vc}}{dt} + \rho_{lc}\frac{dV_{lc}}{dt} + V_{lc}\frac{d\rho_{lc}}{dt} = W_{lec} - W_{vsc}$$

Hacemos las siguientes interpolaciones polinómicas de orden n:

$$\begin{cases} \rho_{vc} = a_0 + a_1 P + a_2 P^2 + \ldots + a_n P^n \\ \rho_{lc} = b_0 + b_1 P + b_2 P^2 + \ldots + b_n P^n \end{cases}$$

Así, cuanto mayor sea el orden de la interpolación, n, mayor será la aproximación que tengamos. Llamamos:

$$\begin{cases} \dfrac{d\rho_{vc}}{dP} = a_1 + 2\,a_2\,P + + \ldots + n\,a_n\,P^{n-1} = k_1 \\ \dfrac{d\rho_{lc}}{dP} = b_1 + 2\,b_2\,P + + \ldots + n\,b_n\,P^{n-1} = k_2 \end{cases} \quad \text{entonces:}$$

$$\begin{cases} \dfrac{d\rho_{vc}}{dt} = \dfrac{d\rho_{vc}}{dP}\dfrac{dP}{dt} = (a_1 + 2\,a_2\,P + \ldots)\dfrac{dP}{dt} = k_1\dfrac{dP}{dt} \\ \dfrac{d\rho_{lc}}{dt} = \dfrac{d\rho_{lc}}{dP}\dfrac{dP}{dt} = (b_1 + 2\,b_2\,P + \ldots)\dfrac{dP}{dt} = k_2\dfrac{dP}{dt} \end{cases} \Longrightarrow$$

$$\left[\rho_{vc}\frac{dV_{vc}}{dt} + V_{vc}\,k_1\frac{dP}{dt} + \rho_{lc}\frac{dV_{lc}}{dt} + V_{lc}\,k_2\frac{dP}{dt} = W_{lec} - W_{vsc}\right]$$

Sabiendo que:

$$\begin{cases} V_c = \text{volumen caldera} = \Pi\,R_c^2\,L_c \\ V_{lc} = \text{volumen líquido saturado en la caldera} = \Pi\,R_c^2\,n_c \\ V_{vc} = \text{volumen vapor saturado en la caldera} = \Pi\,R_c^2\,(L_c - n_c) \end{cases}$$

y que:

$$\frac{dV_{lc}}{dt} = -\frac{dV_{vc}}{dt} = \Pi\,R_c^2\frac{dn_c}{dt} = A_0\frac{dn_c}{dt} \qquad \text{donde llamo: } \left[A_0 = \Pi\,R_c^2\right]$$

entonces:

$$-A_0\,\rho_{vc}\frac{dn_c}{dt} + V_{vc}\,k_1\frac{dP}{dt} + \rho_{lc}\,A_0\frac{dn_c}{dt} + V_{lc}\,k_2\frac{dP}{dt} = W_{lec} - W_{vsc}$$

Llamando: $\left\{\left[\begin{array}{l} A_1 = \rho_{lc} - \rho_{vc} \\ A_2 = V_{lc}\,k_2 + V_{vc}\,k_1 \end{array}\right]\right. \rightarrow \quad A_0\,A_1\dfrac{dn_c}{dt} + A_2\dfrac{dP}{dt} = W_{lec} - W_{vscv} \quad (I)$

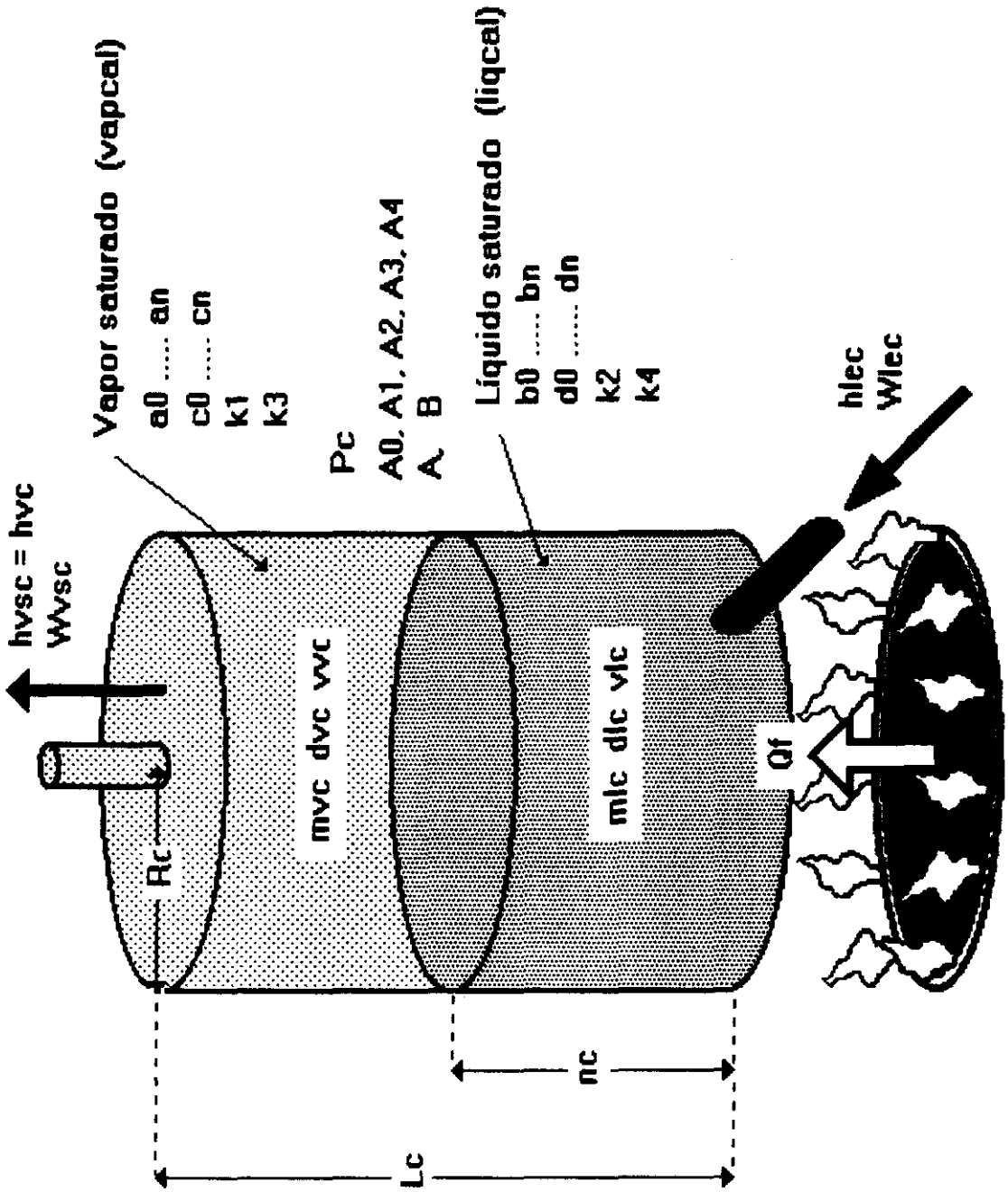

Vapor saturado (vapcal)
a0 ..... an
c0 ..... cn
k1
k3
Pc
A0, A1, A2, A3, A4
A, B
Líquido saturado (liqcal)
b0 ..... bn
d0 ..... dn
k2
k4
hlec
Wlec
hvsc = hvc
Wvsc
mvc dvc vvc
mlc dlc vlc
Qf
Rc
nc
Lc

Figura 5.7

Balance de energía.

La variación de la energía del fluído interior de la caldera es igual a los aportes energéticos del agua suministrada más el del quemador, menos la energía perdida por el vapor saturado que escapa de la caldera:

$$\frac{d}{dt}(m_{vc}\, h_{vc} + m_{lc}\, h_{lc}) = W_{lec}\, h_{lec} - W_{vsc}\, h_{vc} + Q_f$$

$$\begin{cases} m_{vc} = \rho_{vc}\, V_{vc} \\ m_{lc} = \rho_{lc}\, V_{lc} \end{cases} \qquad \text{Haciendo :} \quad \begin{cases} h_{vc} = c_0 + c_1 P + c_2 P^2 + \ldots + c_n P^n \\ h_{lc} = d_0 + d_1 P + c_2 P^2 + \ldots + d_n P^n \end{cases}$$

$$\text{Entonces:} \quad \begin{cases} \frac{dh_{vc}}{dP} = c_1 + 2\, c_2 P + \ldots n\, c_n P^{n-1} = k_3 \\ \frac{dh_{lc}}{dP} = d_1 + 2\, d_2 P + \ldots n\, d_n P^{n-1} = k_4 \end{cases} \rightarrow \quad \begin{cases} \frac{dh_{vc}}{dt} = k_3 \frac{dP}{dt} \\ \frac{dh_{lc}}{dt} = k_4 \frac{dP}{dt} \end{cases}$$

Sustituyendo:

$$h_{vc} \frac{dm_{vc}}{dt} + m_{vc} \frac{dh_{vc}}{dt} + h_{lc} \frac{dm_{lc}}{dt} + m_{lc} \frac{dh_{lc}}{dt} =$$

$$= h_{vc}\, \rho_{vc} \frac{dV_{vc}}{dt} + h_{vc}\, V_{vc} \frac{d\rho_{vc}}{dt} + \rho_{vc}\, V_{vc} \frac{dh_{vc}}{dt} + h_{lc}\, \rho_{lc} \frac{dV_{lc}}{dt} + h_{lc}\, V_{lc} \frac{d\rho_{lc}}{dt} + \rho_{lc}\, V_{lc} \frac{dh_{lc}}{dt}$$

$$= -h_{vc}\, \rho_{vc}\, A_0 \frac{dn_c}{dt} + h_{vc}\, V_{vc}\, k_1 \frac{dP}{dt} + \rho_{vc}\, V_{vc}\, k_3 \frac{dP}{dt} + h_{lc}\, \rho_{lc}\, A_0 \frac{dn_c}{dt} + h_{lc}\, V_{lc}\, k_2 \frac{dP}{dt} +$$

$$+ \rho_{lc}\, V_{lc}\, k_4 \frac{dP}{dt} =$$

$$= A_0 \frac{dn_c}{dt} (h_{lc}\, \rho_{lc} - h_{vc}\, \rho_{vc}) + \frac{dP}{dt} (k_1\, h_{vc}\, V_{vc} + k_2\, h_{lc}\, V_{lc} + k_3\, \rho_{vc}\, V_{vc} + k_4\, \rho_{lc}\, V_{lc})$$

Llamando:

$$\left\{\begin{bmatrix} A_3 = h_{lc}\,\rho_{lc} - h_{vc}\,\rho_{vc} \\ A_4 = k_1\,h_{vc}\,V_{vc} + k_2\,h_{lc}\,V_{lc} + k_3\,\rho_{vc}\,V_{vc} + k_4\,\rho_{lc}\,V_{lc} \end{bmatrix}\right. \rightarrow$$

$$A_0\,A_3\,\frac{dn_c}{dt} + A_4\,\frac{dP}{dt} = W_{lec}\,h_{lec} - W_{vsc}\,h_{vc} + Q_f \qquad (II)$$

Ecuaciones finales:

$$\begin{cases} A_0\,A_1\,\dfrac{dn_c}{dt} + A_2\,\dfrac{dP}{dt} = W_{lec} - W_{vsc} \\ A_0\,A_3\,\dfrac{dn_c}{dt} + A_4\,\dfrac{dP}{dt} = W_{lec}\,h_{lec} - W_{vsc}\,h_{vc} + Q_f \end{cases} \rightarrow$$

$$\frac{dn_c}{dt} = \frac{W_{vsc}\left(\frac{A_4}{A_2} - h_{vc}\right) - W_{lec}\left(\frac{A_4}{A_2} - h_{lec}\right) + Q_f}{A_3\,A_0 - \frac{A_0\,A_1\,A_4}{A_2}} \qquad (A)$$

$$\frac{dP}{dt} = \frac{W_{vsc}\left(\frac{A_3}{A_1} - h_{vc}\right) - W_{lec}\left(\frac{A_3}{A_1} - h_{lec}\right) + Q_f}{A_4 - \frac{A_3\,A_2}{A_1}} \qquad (B)$$

Resumiendo las expresiones más importantes:

$$\begin{cases} A_0 = \Pi\, R_c^2 \\ A_1 = \rho_{lc} - \rho_{vc} \\ A_2 = V_{lc}\, k_2 + V_{vc}\, k_1 \\ A_3 = h_{lc}\, \rho_{lc} - h_{vc}\, \rho_{vc} \\ A_4 = k_1\, h_{vc}\, V_{vc} + k_2\, h_{lc}\, V_{lc} + k_3\, \rho_{vc}\, V_{vc} + k_4\, \rho_{lc}\, V_{lc} \end{cases}$$

$$\begin{cases} k_1 = a_1 + 2\, a_2\, P + \ldots + n\, a_n\, P^{n-1} \\ k_2 = b_1 + 2\, b_2\, P + \ldots + n\, b_n\, P^{n-1} \\ k_3 = c_1 + 2\, c_2\, P + \ldots + n\, c_n\, P^{n-1} \\ k_4 = d_1 + 2\, d_2\, P + \ldots + n\, d_n\, P^{n-1} \end{cases} \qquad \begin{cases} \rho_{vc} = a_0 + a_1\, P + a_2\, P^2 + \ldots + a_n\, P^n \\ \rho_{lc} = b_0 + b_1\, P + b_2\, P^2 + \ldots + b_n\, P^n \\ h_{vc} = c_0 + c_1\, P + c_2\, P^2 + \ldots + c_n\, P^n \\ h_{lc} = d_0 + d_1\, P + c_2\, P^2 + \ldots + d_n\, P^n \end{cases}$$

El cálculo de los coeficientes a1, ..., an, b1, ..., bn, c1, ..., cn y d1, ..., dn se realizará interpolando las curvas correspondientes a los estados de líquido saturado y vapor saturado. Remitimos desde aquí al apartado V.3, donde serán descritos pormenorizadamente, entre otros, estos cálculos.

MODELO ORIENTADO A OBJETO.

Crearemos la clase *Caldera*, con los siguientes métodos:

— *Protegidos:*

Dos objetos de las clases Líquido_saturado y Vapor_saturado serán los fluídos interiores de la caldera. Una serie de variables representarán los caudales, dimensiones, magnitudes termodinámicas, etc. También habrá una serie de variables para calcular las ecuaciones desarrolladas en el modelo teórico.

— *Públicos:*

Mediante el constructor, asignaremos los valores fijos de la altura y el radio, y los valores iniciales del nivel y la presión; mediante otra función, asignaremos los parámetros iniciales, y con otra efectuaremos los cálculos para cada instante. Otras devolverán estos valores.

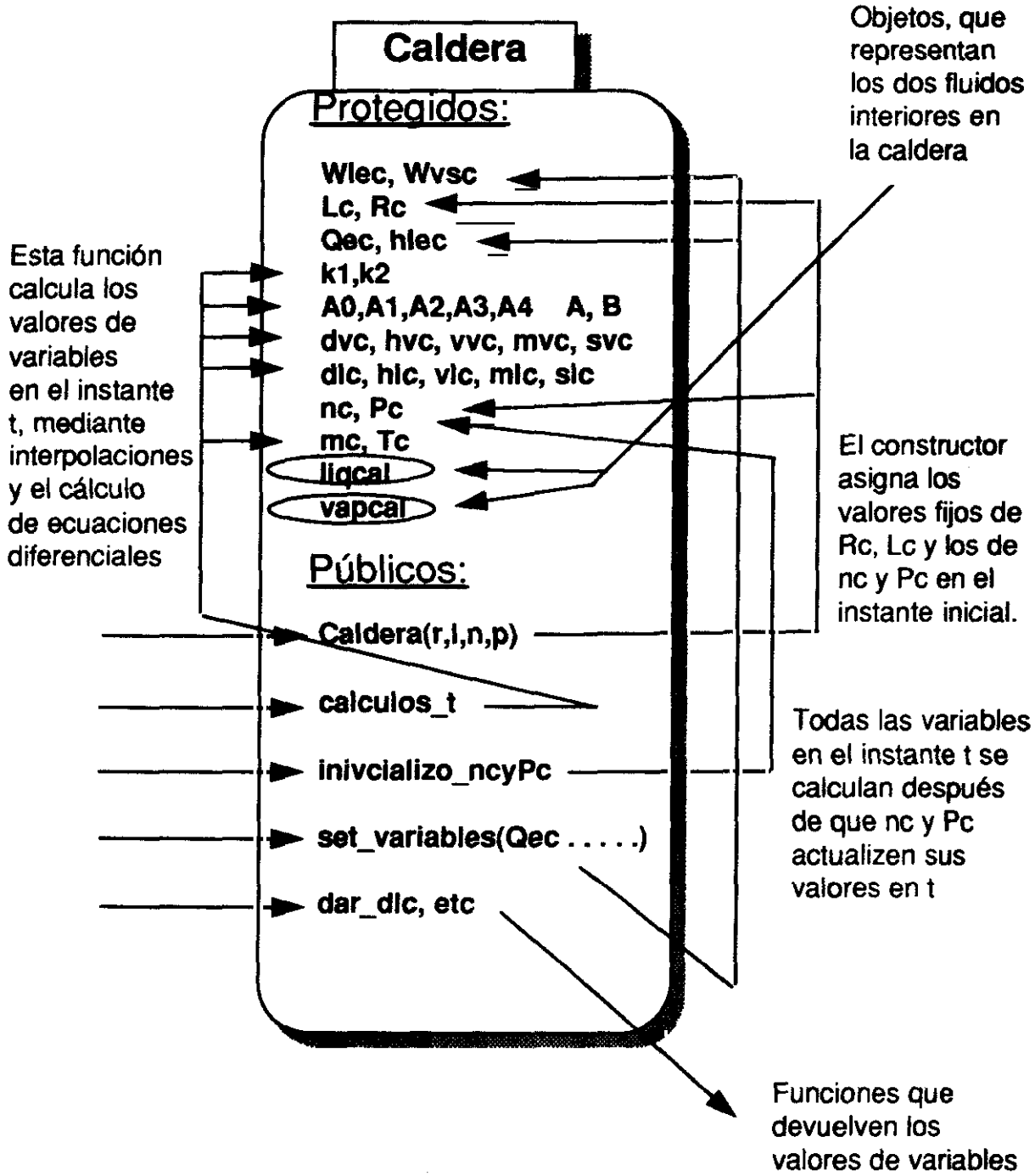

Caldera
Protegidos:
Wlec, Wvsc
Lc, Rc
Qec, hlec
k1,k2
A0,A1,A2,A3,A4 A, B
dvc, hvc, vvc, mvc, svc
dlc, hlc, vlc, mlc, slc
nc, Pc
mc, Tc
liqcal
vapcal
Públicos:
Caldera(r,l,n,p)
calculos_t
inivcializo_ncyPc
set_variables(Qec . . . . .)
dar_dlc, etc
Objetos, que representan los dos fluidos interiores en la caldera
Esta función calcula los valores de variables en el instante t, mediante interpolaciones y el cálculo de ecuaciones diferenciales
El constructor asigna los valores fijos de Rc, Lc y los de nc y Pc en el instante inicial.
Todas las variables en el instante t se calculan después de que nc y Pc actualizen sus valores en t
Funciones que devuelven los valores de variables
= objeto

Figura 5.8

V.2.8.- SUPERCALENTADOR.-

MODELO TEORICO.

En la figura 5.9 podemos observar esquematizado el modelo del supercalentador. Consiste en un habitáculo que recibe un aporte energético de un quemador, de valor Q_{es}. Recibe también un flujo W_{ves} de vapor saturado (proveniente de la caldera, de propiedades P_{ves}, T_{ves}, s_{ves}, h_{ves} y v_{ves}) y de él sale un flujo de vapor (P_{vss}, T_{vss}, h_{vss}, s_{vss}, v_{vss}, x_{vss}, W_{vss}, que podrá ser vapor normal ó supercalentado), a un caudal W_{vss} determinado por una válvula a la salida, y que vamos a suponer que, en todo instante, permanece al mismo valor que el caudal de entrada. En su interior habrá una cantidad de masa de vapor m_{vs} que supondremos en todo momento de valor constante:

$$\begin{cases} W_{ves} = W_{vss} \\ m_{vs} = cte. \end{cases}$$

La diferencia de presión entre la salida y la entrada será directamente proporcional al cuadrado del caudal de entrada e inversamente proporcional a la densidad del vapor de entrada; esta diferencia suele ser muy pequeña, pudiendo aproximarse la mayoría de las veces que $P_{ves} = P_{vss}$, pues el segundo término de la siguiente relación es, en órdenes de magnitud, muy pequeño:

$$P_{ves} - P_{vss} = k \frac{W_{ves}^2}{\rho_{ves}} \approx 0$$

Por otro lado, haciendo un balance de energías, tenemos que la variación energética del vapor interior en el supercalentador es igual a la energía recibida tanto por el vapor de entrada como por el aporte del quemador, menos la energía perdida por el vapor supercalentado que escapa:

$$m_{vs} \frac{dh_{vss}}{dt} = Q_{es} + W_{ves} (h_{ves} - h_{vss})$$

Esta última ecuación es diferencial de 1er orden. Resolviéndola:

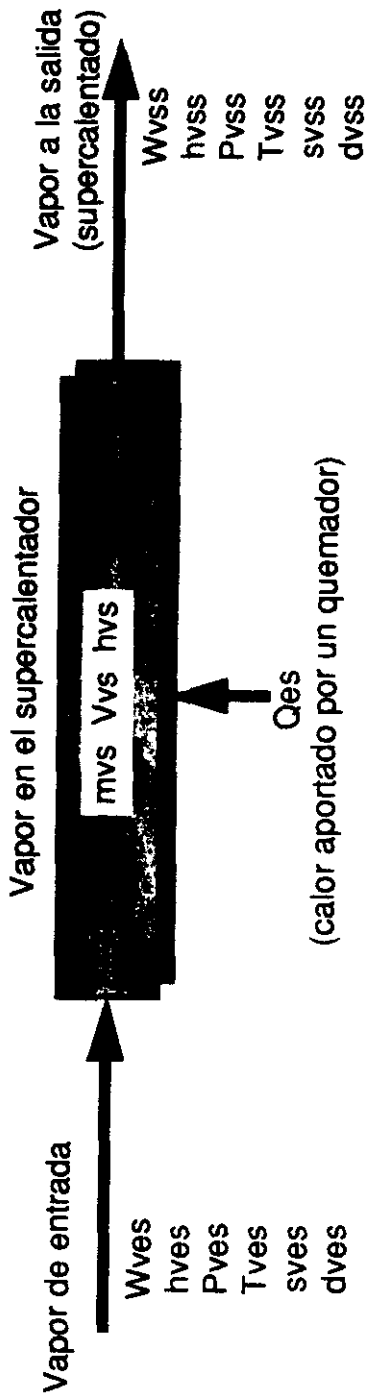

Vapor a la salida
(supercalentado)
Wvss
hvss
Pvss
Tvss
svss
dvss
Vapor en el supercalentador
mvs Vvs hvs
Qes
(calor aportado por un quemador)
Vapor de entrada
Wves
hves
Pves
Tves
sves
dves

Figura 5.9

$$\frac{dh_{vss}}{dt} + \frac{W_{ves}}{m_{vs}} h_{vss} - \frac{Q_{es} + W_{ves}\, h_{ves}}{m_{vs}} = 0 \quad \text{Llamando:} \begin{cases} A = 1 \\ B = \dfrac{W_{ves}}{m_{vs}} \\ C = -\dfrac{(Q_{es} + W_{ves}\, h_{ves})}{m_{vs}} \\ y = h_{vss} \\ x = t \end{cases} \rightarrow$$

$$\begin{cases} A\, y' + B\, y + C = 0 \\ y(x_0) = x_0 \end{cases} \rightarrow \quad y(x) = -\frac{C}{B} + k\, e^{-(\frac{B}{A} x)} \quad \text{donde} \quad k = (y_0 + \frac{C}{B})\, e^{(\frac{B}{A} x_0)}$$

Por tanto:

$$h_{vss} = \frac{Q_{es} + h_{ves}\, W_{ves}}{W_{ves}} + k\, e^{(-\frac{W_{ves}}{m_{vs}} t)}$$

donde:

$$k = \left(h_{vss}(t_0) - \frac{Q_{es} + h_{ves}\, W_{ves}}{W_{ves}} \right) e^{(\frac{W_{ves}}{m_{vs}} t_0)}$$

MODELO ORIENTADO A OBJETO.

Ver figura 5.10. La clase es *Supercalentador*, y consta de los siguientes miembros:

— *Protegidos:*

Un conjunto de variables representarán las propiedades termodinámicas de los estados de vapor a la entrada y a la salida del supercalentador; otras serán las características propias del mismo, y una hará referencia al aporte del quemador. Un objeto de la clase *Vapor* se encargará de representar el estado de salida.

— *Públicos:*

El constructor, además de crear el objeto, inicializará los valores de kx, hvss y ksup. Un método se encargará de asignar valores al estado de vapor de entrada, y otro realizará una serie de cálculos, consistentes tanto en la resolución de la ecuación diferencial como en el cálculo de las magnitudes del estado del vapor a la salida (supercalentado o no). Una serie de funciones devolverán los valores de los estados.

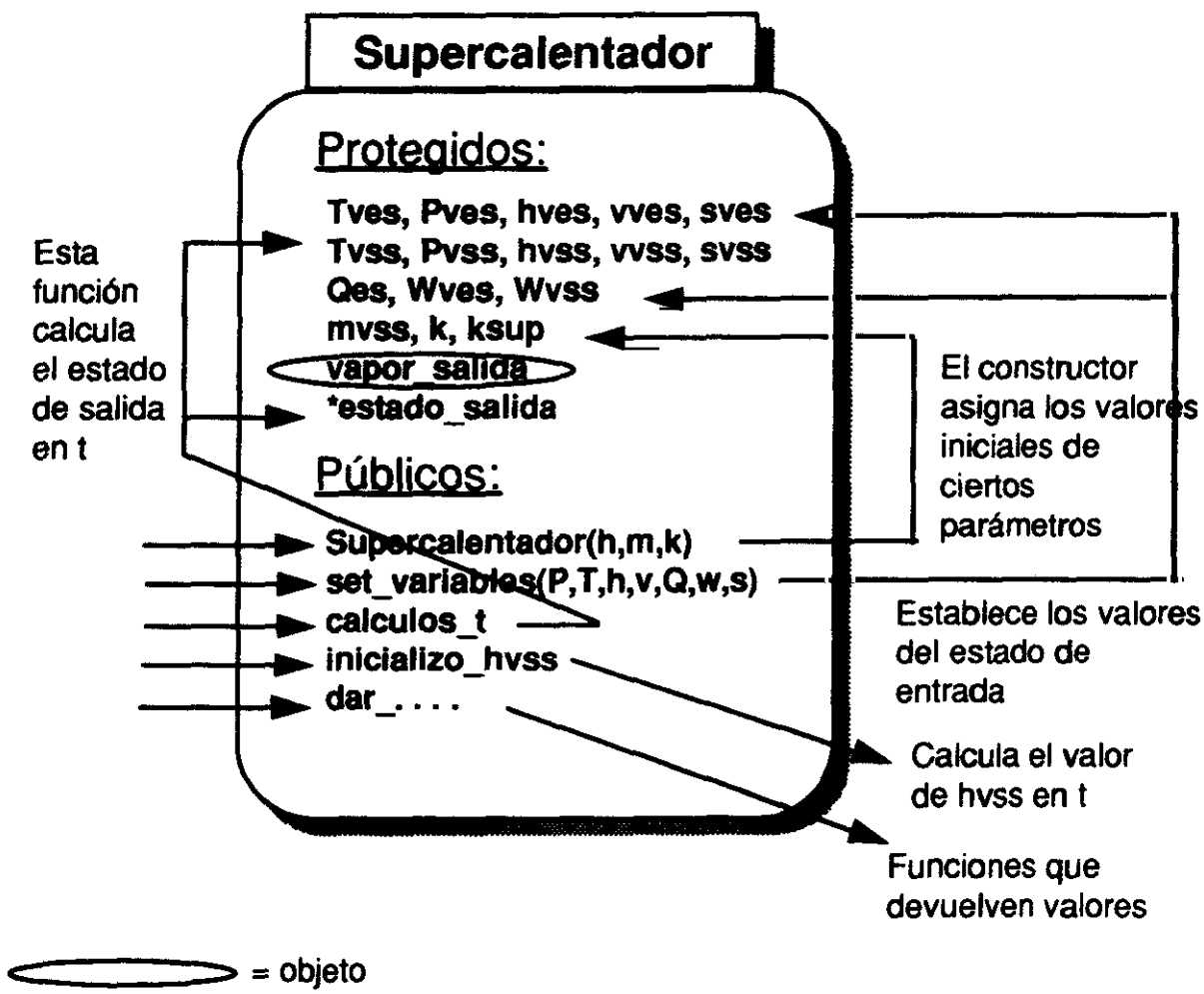

Supercalentador
Protegidos:
Tves, Pves, hves, vves, sves
Tvss, Pvss, hvss, vvss, svss
Qes, Wves, Wvss
mvss, k, ksup
vapor_salida
*estado_salida
Públicos:
Supercalentador(h,m,k)
set_variables(P,T,h,v,Q,w,s)
calculos_t
inicializo_hvss
dar_. . . .
Esta función calcula el estado de salida en t
El constructor asigna los valores iniciales de ciertos parámetros
Establece los valores del estado de entrada
Calcula el valor de hvss en t
Funciones que devuelven valores
= objeto

Figura 5.10

V.2.9.- TURBINA.-

MODELO TEORICO.

Vamos a concebir nuestra turbina como un dispositivo de una entrada y dos salidas (ver Fig. 5.11). Por la entrada introducimos vapor a un caudal W_{vet} (P_{vet}, h_{vet}, T_{vet}, s_{vet}, v_{vet} y h_{vet}), parte del cual será expulsado a través de una salida (a razón de W_{tx} m^3 por segundo) y el resto abandonará la turbina para continuar su camino a través de la planta, con un caudal W_{vst} que será la diferencia del de entrada menos el de expulsión, y con unas nuevas propiedades P_{vst}, T_{vst}, h_{vst}, s_{vst}, v_{vst} y s_{vst}.

En una turbina, el vapor de entrada es expandido al pasar por ella, disminuyendo su presión, entalpía y temperatura, y aumentando su volumen, de forma que se produce un trabajo útil que es posible aprovechar (este trabajo será la diferencia de entalpías de salida y entrada, y no interviene el gas de expulsión, pues vamos a suponer que el proceso de expansión es adiabático, esto es, sin pérdidas de calor con el exterior), y es descrito en términos de un rendimiento isoentnrópico η_i, el cual es a su vez función del flujo del vapor de entrada.

Vamos a considerar las siguientes relaciones para construir nuestro modelo. Las que sean representaciones de cálculos de tablas de vapor, serán más detalladas en el apartado V.3.

(1) Flujo salida = flujo entrada - flujo expulsión:

$$W_{vst} = W_{vet} - W_{tx}$$

(2) El flujo expulsado es una fracción del de entrada:

$$W_{tx} = \Phi(W_t) = k_x W_t$$

(3) Rendimiento isoentrópico, es función del flujo de entrada:

$$\eta_i = \Phi(W_t)$$

Vamos a considerar, de momento, que este rendimiento es constante, de valor 0.5.

(4) Mediante las tablas de vapor vamos a calcular la entalpía h_{vsti}, que es la h_{vst} en el caso de un proceso isoentrópico, pues

$$s_{vst} = s_{vet}: \quad h_{vsti} = \Phi(P_{vst}, s_{vet})$$

(5) Calculamos la h_{vst} real (no isoentrópica):

$$h_{vst} = h_{vsti} - \eta_i (h_{vet} - h_{vsti})$$

(6) Mediante las tablas de vapor, hallamos:

$$\begin{cases} T_{vst} = \Phi (P_{vst}, h_{vst}) \\ s_{vst} = \Phi (P_{vst}, h_{vst}) \\ \rho_{vst} = \Phi (P_{vst}, h_{vst}) \end{cases}$$

(7) Por último, la potencia generada por la turbina será:

$$\Pi_t = W_t (h_{vet} - h_{vst})$$

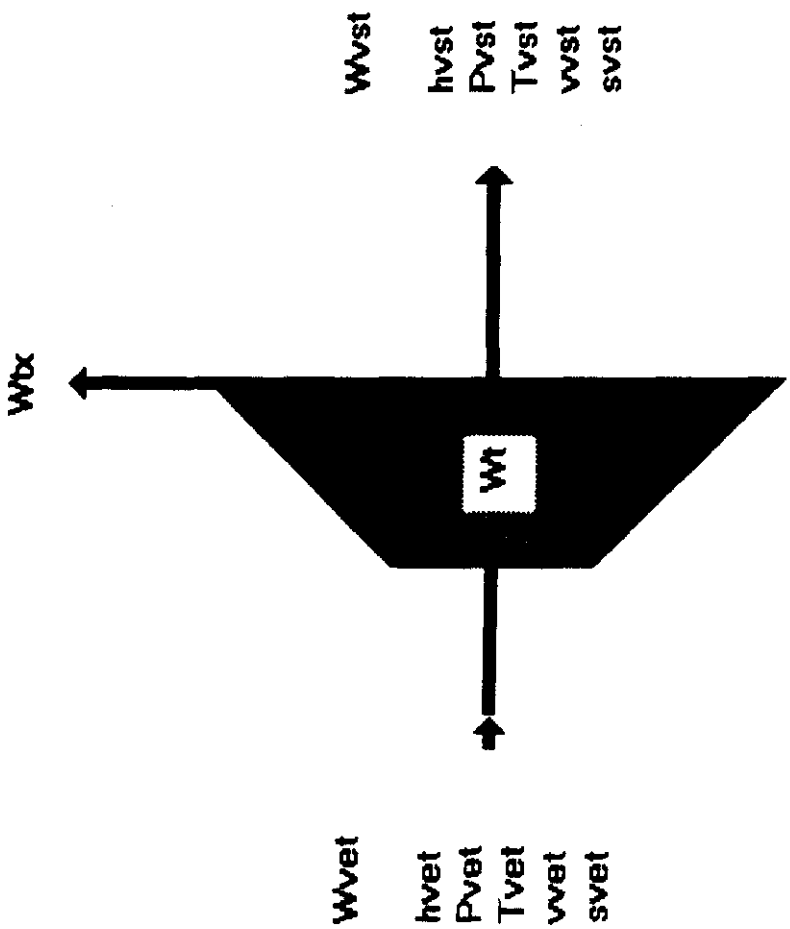

Wvst
hvst
Pvst
Tvst
vvst
svst
Wtx
Wt
Wvet
hvet
Pvet
Tvet
vvet
svet

Figura 5.11

MODELO ORIENTADO A OBJETO.

En la figura 5.12 puede verse esquematizada la clase Turbina, que consta de los siguientes miembros:

— *Protegidos:*

Un grupo de variables representará las propiedades termodinámicas del vapor a la entrada (W_{vet}, P_{vet}, T_{vet}, h_{vet}, s_{vet}, v_{vet}), y otro grupo las del vapor de salida (W_{vst}, P_{vst}, T_{vst}, h_{vst}, s_{vst}, v_{vst}); otras variables serán las características de la turbina (W_{tx}, k_x) y otras variables necesarias para los cálculos (R_i, h_{vsti}).Dos objetos de la clase vapor representarán a los estados de salida real e ideal, y un método calculará todas las magnitudes y resolverá las ecuaciones para cada instante de la simulación.

— *Públicos:*

El constructor inicializará el valor de kx; una función asignará los valores de las variables del estado a la entrada, de la presión a la salida, y calculará el otro estado. Unos métodos controlarán el valor de k_x, mientras que otros retornarán los valores de las variables.

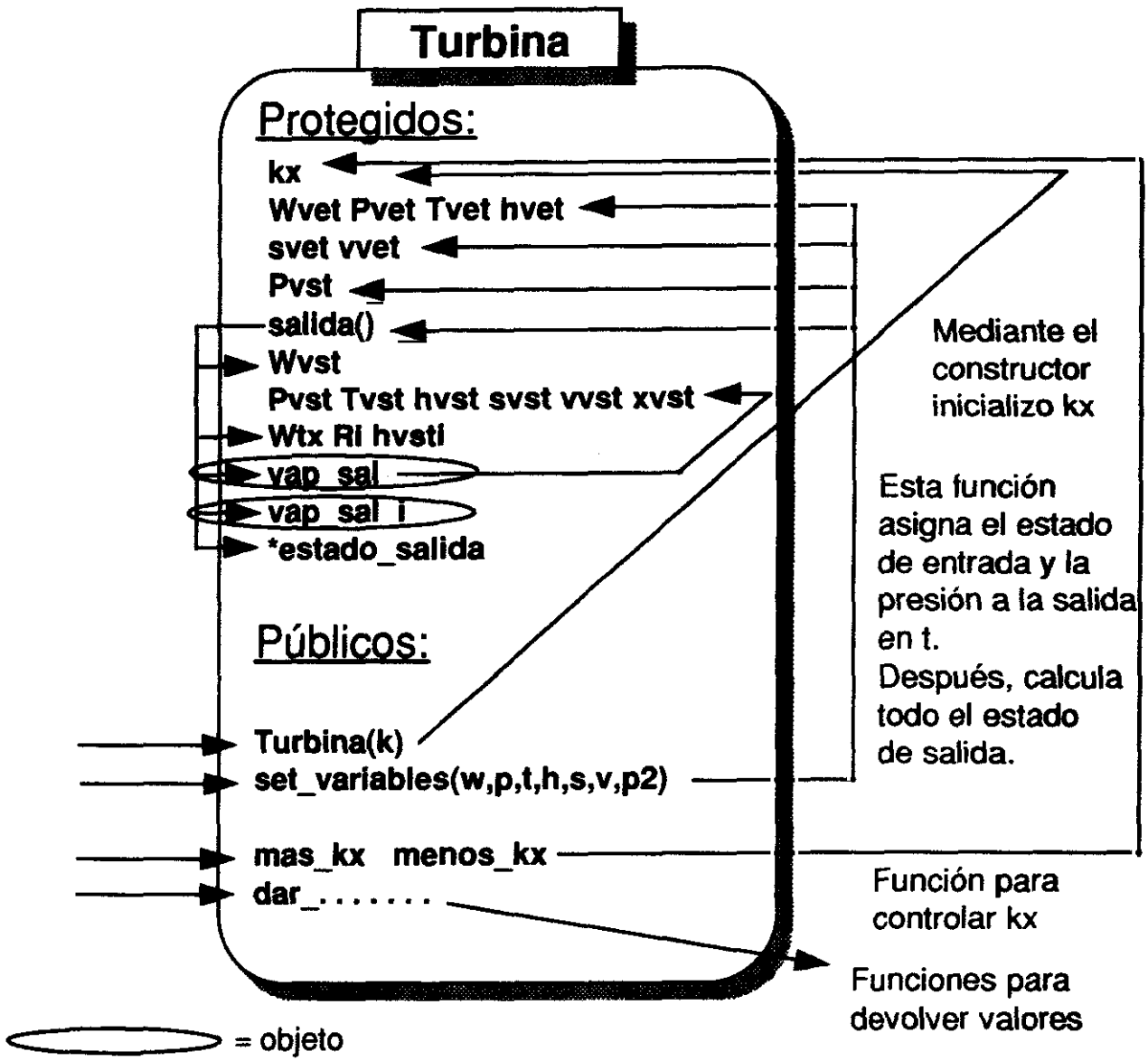

Turbina
Protegidos:
kx
Wvet Pvet Tvet hvet
svet vvet
Pvst
salida()
Wvst
Pvst Tvst hvst svst vvst xvst
Wtx RI hvsti
vap_sal
vap_sal_i
*estado_salida
Públicos:
Turbina(k)
set_variables(w,p,t,h,s,v,p2)
mas_kx menos_kx
dar_.....
Mediante el constructor inicializo kx
Esta función asigna el estado de entrada y la presión a la salida en t.
Después, calcula todo el estado de salida.
Función para controlar kx
Funciones para devolver valores
= objeto

Figura 5.12

V.2.10.- CONDENSADOR.-

MODELO TEORICO.

El condensador es un dispositivo cuya tarea consiste en hacer pasar el vapor a su estado líquido (proceso de condensación: vapor -> vapor saturado -> líquido saturado). Vamos a considerarlo como un habitáculo cilíndrico de radio R_{cd} y altura L_{cd}, con una entrada para el vapor y una salida para el líquido. Un sistema de refrigeración, consistente en unos conductos que rodean el condensador y por los cuales circula agua, será necesario para llevar a cabo el proceso de la condensación. Nosotros supondremos el circuito del refrigerante como una sola tubería por cuyo interior circula agua, y que, bien entrando o rodeando al condensador, producirá el intercambio energético requerido.

Vamos a considerar en nuestro modelo el conjunto condensador-sistema de refrigeración, que vemos esquematizado en la figura 5.13. El condensador recibe en su entrada el vapor proveniente, por ejemplo, de la turbina, a un flujo W_{vecd}, y con valores h_{vecd} y d_{vecd}. Hasta llegar al estado de vapor saturado, va pasando por estados intermedios, de manera que, a efectos de modelado, consideramos que en el interior del condensador hay un vapor con valores m_{vcd}, d_{vcd} y V_{vcd}, y que luego calcularemos teniendo en cuenta que tenemos una mezcla de vapor de entrada, vapor saturado y estados intermedios. La presión en el interior del condensador será P_{cd}, presión de saturación, y el nivel de líquido será n_{cd}.

El líquido formado estará saturado, y tendrá unos valores m_{lcd}, d_{lcd} y V_{lcd}. Puesto que este líquido que reside en el interior del condensador es el mismo que el que sale (líquido de salida), podemos decir que:

$$\begin{cases} h_{lscd} = h_{lcd} \\ \rho_{lscd} = \rho_{lcd} \end{cases}$$

El líquido de salida abandonará el condensador a un caudal fijado por una bomba (que estará a continuación del condensador y que bombea, entre otras razones, para subir posteriormente la presión del agua para que no sea ya líquido saturado), de valor W_{lscd}, y consideraremos la entalpía de salida h_{lscd}.

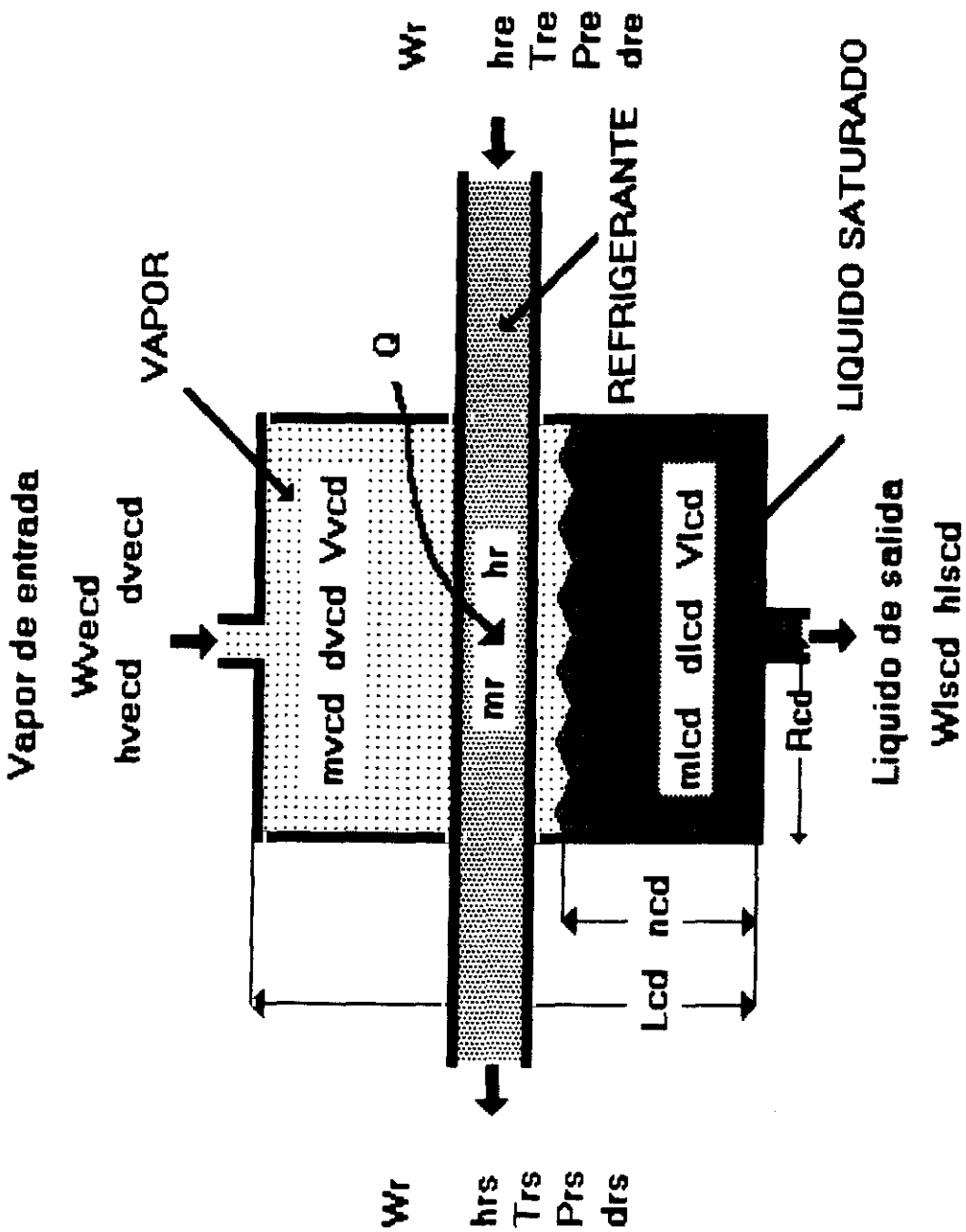

Wr
hre
Tre
Pre
dre
REFRIGERANTE
LIQUIDO SATURADO
VAPOR
Q
Vapor de entrada
Wvecd
hvecd
dvecd
mvcd dvcd Vvcd
mr hr
mlcd dlcd Vlcd
Rcd
Liquido de salida
Wlscd hlscd
ncd
Lcd
Wr
hrs
Trs
Prs
drs

Figura 5.13

En cuanto al sistema de refrigeración, lo supondremos como una sola tubería que penetra o rodea al condensador, aislada del mismo excepto en su aspecto térmico, y por la que circula agua. Por ella entrará agua mediante un caudal W_r, y con unas características h_{re}, T_{re} y P_{re}, saliendo agua al mismo caudal, pero con valores distintos, h_{rs}, P_{rs} y T_{rs}, debido al intercambio energético producido. A efectos del cálculo de esta transferencia de calor, consideraremos que la cantidad de agua refrigerante implicada en este proceso es de una masa m_r y con una entalpía h_r.

En la elaboración de nuestro modelo desglosaremos los distintos subsistemas del conjunto caldera - refrigerante. Hemos tenido en cuenta, además de las anteriores consideraciones, una serie de supuestos teóricos, que a continuación damos:

Supuesto 1.

— Introducción:

El vapor que hay en el interior del condensador es una mezcla de vapores de distintas calidades. Partiendodel vapor de entrada (v_{ecd}) hasta llegar a líquido saturado, el vapor pasa por distintos estados. En la simulación del modelo es farragoso el tener que establecer todos y cada uno de los distintos vapores, lo cual haría necesario computar numerosas ecuaciones, puesto que tendríamos que tener en cuenta la cantidad de masa de cada tipo de vapor:

$$m_{vcd} = \sum_{i} m_{vcdi}$$

donde i representa a todos los vapores, desde $x = x_{vecd}$ (vapor de entrada) hasta $x = 0$ (vapor saturado). Ver la figura 5.14.

— Enunciado:

Consideraremos que el vapor que hay en el interior de la caldera es, en todo momento, una mezcla de dos vapores (que ocupan el mismo volumen, V_{cd}, y están a la misma presión, P_{cd}), y que son (ver figura 5.14):

$$\text{Vapor } \mathbf{vecd}: \; h_{vecd} \; ; \; \begin{cases} P_{cd} \\ h_{vecd} \end{cases} \rightarrow T_{vecd} \, , \, \rho_{vecd} \, , \, x_{vecd}$$

$m_{vecd} = V_{vcd} \cdot d_{vecd}$ = masa dentro del condensador del vapor de la misma calidad del vapor de entrada.

Vapor **m** : $h_m = ¿?$ Impongo que $x_m = \frac{x_{vecd}}{2}$, luego:

$$\begin{cases} P_{cd} \\ x_m \end{cases} \rightarrow T_m , \rho_m , x_m \qquad m_m = V_{vcd} \cdot d_{vecd}$$

Supuesto 2.

Este supuesto hace referencia al refrigerante. Vamos a suponer que:

— $P_{re} = P_{rs} = P_r$, las presiones del agua refrigeradora a la entrada y a la salida tienen el mismo valor.

— $W_{re} = W_{rs} = W_r$, el caudal es constante.

— $P_r = P_{cd}$. La presión del agua en el interior del sistema de refrigeración es la que tenga el condensador.

— $h_r = h_{rs}$. Esto significa que vamos a suponer que el intercambio energético se realiza para una entalpía que consideramos es la de salida.

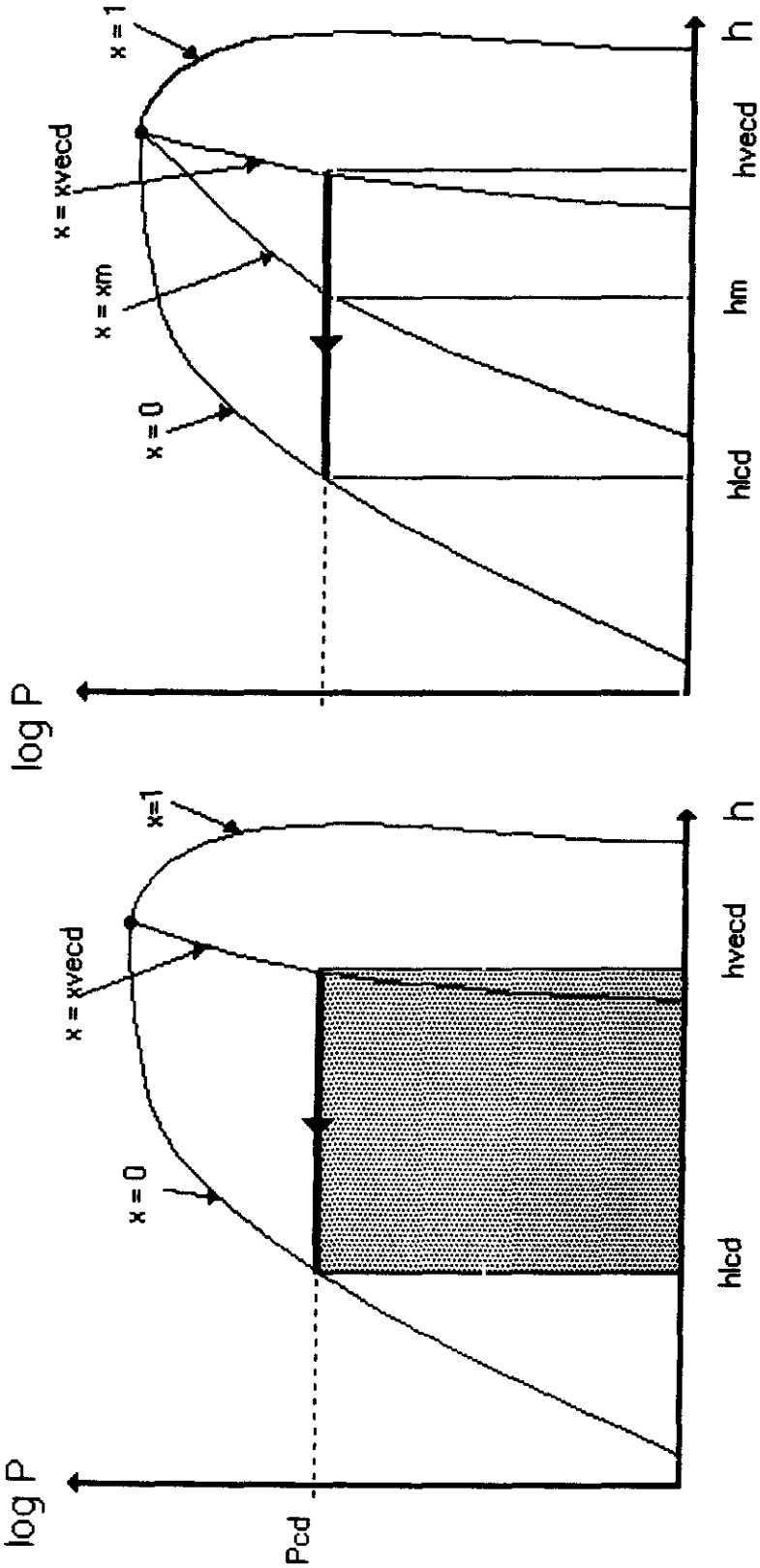

x = 1
x = xvecd
x = xm
x = 0
h
hvecd
hm
hlcd
log P
x=1
x = xvecd
x = 0
h
hvecd
hlcd
log P
Pcd

Figura 5.14

Desarrollo del modelo.

A continuación, desarrollamos el aspecto matemático del modelo:

Tenemos las siguientes relaciones:

- $\begin{cases} m_{vcd} = m_{vecd} + m_m = \rho_{vecd} V_{vcd} + \rho_m V_{vcd} = (\rho_{vecd} + \rho_m) V_{vcd} \\ m_{lcd} = \rho_{lcd} V_{lcd} \end{cases}$

- $V_{cd} = V_{lcd} + V_{vcd} = \Pi R_{cd}^2 L_{cd}$ $\begin{cases} V_{vcd} = \Pi R_{cd}^2 (L_{cd} - n_{cd}) \\ V_{lcd} = \Pi R_{cd}^2 n_{cd} \end{cases}$

- $\frac{dV_{lcd}}{dt} = -\frac{dV_{vcd}}{dt} = \Pi R_{cd}^2 \frac{dn_{cd}}{dt}$

- lcd = líquido saturado $\rightarrow$ $\begin{cases} \rho_{lcd} = b_0 + b_1 P_{cd} + ... + b_n P_{cd}^n \\ h_{lcd} = d_0 + d_1 P_{cd} + ... + d_n P_{cd}^n \\ \frac{d\rho_{lcd}}{dP_{cd}} = b_1 + ... + n\, b_n P_{cd}^{n-1} = k_2 \\ \frac{dh_{lcd}}{dP_{cd}} = d_1 + ... + n\, d_n P_{cd}^{n-1} = k_4 \\ \frac{dh_{lcd}}{dt} = k_4 \frac{dP_{cd}}{dt} \\ \frac{d\rho_{lcd}}{dt} = k_2 \frac{dP_{cd}}{dt} \end{cases}$

- Refrigerante:

En la figura 5.15 podemos ver el esquema del sistema de refrigeración. Nuestros supuestos dicen:

$$P_{re} = P_{rs} = P_r \quad , \quad W_{re} = W_{rs} = W_r \quad , \quad P_r = P_{cd} \quad y \quad h_r = h_{rs}$$

Balance de masa :

$$\frac{dm_r}{dt} = W_r - W_r = 0 \rightarrow m_r = cte.$$

Balance de energía :

$$\frac{d(m_r\, h_r)}{dt} = W_r\, h_{re} - W_r\, h_{rs} + Q \quad \rightarrow$$

$$\left[m_r \frac{dh_r}{dt} = W_r\, (h_{re} - h_{rs}) + Q \right]$$

donde Q es la energía que cede el conjunto líquido + vapor interiores del condensador al líquido de refrigeración.

Como suponemos que :

$$h_r = h_{rs} \quad \rightarrow \quad m_r \frac{dh_{rs}}{dt} = W_r\, (h_{re} - h_{rs}) + Q \quad \rightarrow$$

$$\left[Q = m_r \frac{dh_{rs}}{dt} - W_r\, (h_{re} - h_{rs}) \right]$$

- Conjunto refrigerante + materia interior del condensador :

[Q = Q']

donde Q' es la energía que cede el refrigerante al interior del condensador, y que obviamente es igual a la que recibe el último de éste, con signo cambiado.

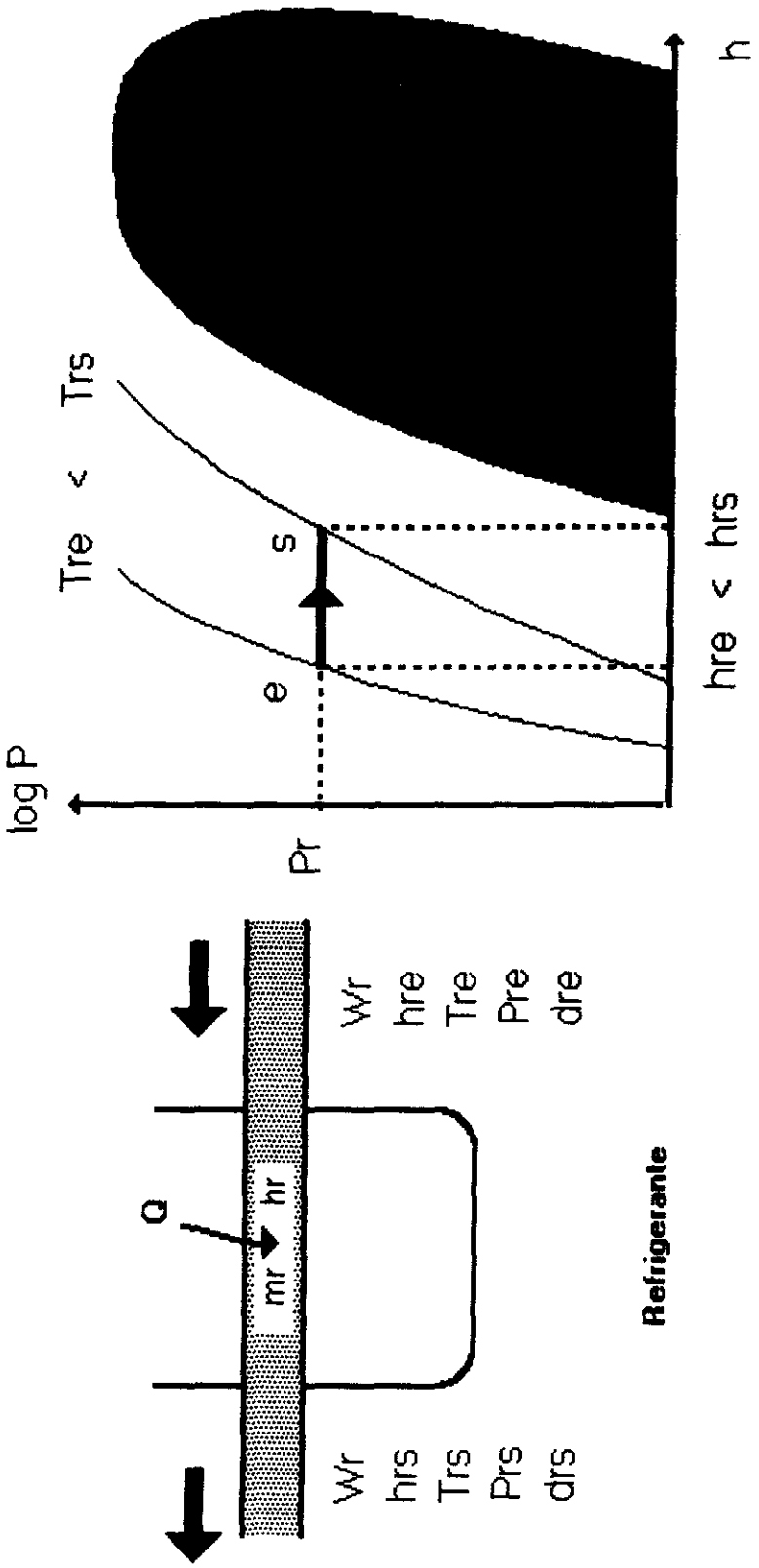

h
Trs
Tre < Trs
s
e
hre < hrs
log P
Pr
Wr
hre
Tre
Pre
dre
Q
hr
mr
Refrigerante
Wr
hrs
Trs
Prs
drs

Figura 5.15

- Materia interior del condensador (ver fig. 5.16) :

Balance de masa:

$$\begin{cases} \left[\dfrac{dm_{cd}}{dt} = W_{vecd} - W_{lscd}\right] \\ m_{cd} = m_{lcd} + m_{vcd} \end{cases} \rightarrow \frac{dm_{lcd}}{dt} + \frac{dm_m}{dt} + \frac{dm_{vecd}}{dt} = W_{vecd} - W_{lscd}$$

Balance de energía:

$$\left[\frac{d}{dt}\left(m_{lcd}\, h_{lcd} + m_{vecd}\, h_{vecd} + m_m\, h_m\right) = W_{vecd}\, h_{vecd} - W_{lscd}\, h_{lscd} + Q'\right]$$

- Estado vapor (ver fig 5.16) :

Para una calidad x dada:

$$\begin{cases} \rho = \Phi(P) = a_0 + a_1 P + \ldots + a_n P^n \\ h = \Phi(P) = c_0 + c_1 P + \ldots + c_n P^n \end{cases} \quad \text{Luego:} \rightarrow$$

$$\begin{cases} \dfrac{d\rho}{dP} = a_1 + 2\, a_2 P + \ldots + n\, a_n P^{n-1} = k_1 \\ \dfrac{dh}{dP} = c_1 + 2\, c_2 P + \ldots + n\, c_n P^{n-1} = k_3 \end{cases} \rightarrow \begin{cases} \dfrac{d\rho}{dt} = k_1 \dfrac{dP}{dt} \\ \dfrac{dh}{dt} = k_3 \dfrac{dP}{dt} \end{cases}$$

Si variamos x, cambian a_i y c_i, y varían los parámetros k_1 y k_3 .

Hemos dado por supuesto que en el interior del condensador tenemos dos vapores, con las notaciones "vecd" y "m", de calidades distintas. Luego:

$$\begin{cases} \rho_{vecd} = \Phi_1(P_{cd}) = a_{01} + a_{11} P_{cd} + \ldots + a_{n1} P_{cd}^n \\ \dfrac{d\rho_{vecd}}{dP_{cd}} = a_{11} + \ldots + n\, a_{n1} P_{cd}^{n-1} = k_{11} \\ \dfrac{d\rho_{vecd}}{dt} = k_{11} \dfrac{dP_{cd}}{dt} \end{cases} \quad \begin{cases} h_{vecd} = c_{01} + c_{11} P_{cd} + \ldots + c_{n1} P_{cd}^n \\ \dfrac{dh_{vecd}}{dP_{cd}} = c_{11} + \ldots + n\, c_{n1} P_{cd}^{n-1} = k_{31} \\ \dfrac{dh_{vecd}}{dt} = k_{31} \dfrac{dP_{cd}}{dt} \end{cases}$$

$$\begin{cases} \rho_m = \Phi_2(P_{cd}) = a_{02} + a_{12} P_{cd} + \ldots + a_{n2} P_{cd}^n \\ \dfrac{d\rho_m}{dP_{cd}} = a_{12} + \ldots + n\, a_{n2} P_{cd}^{n-1} = k_{12} \\ \dfrac{d\rho_m}{dt} = k_{12} \dfrac{dP_{cd}}{dt} \end{cases} \quad \begin{cases} h_m = c_{02} + c_{12} P_{cd} + \ldots + c_{n2} P_{cd}^n \\ \dfrac{dh_m}{dP_{cd}} = c_{12} + \ldots + n\, c_{n2} P_{cd}^{n-1} = k_{32} \\ \dfrac{dh_m}{dt} = k_{32} \dfrac{dP_{cd}}{dt} \end{cases}$$

Por tanto, a_{ij}, c_{ij} y k_{ij} varían con x (la cual, a su vez, cambia con el tiempo).

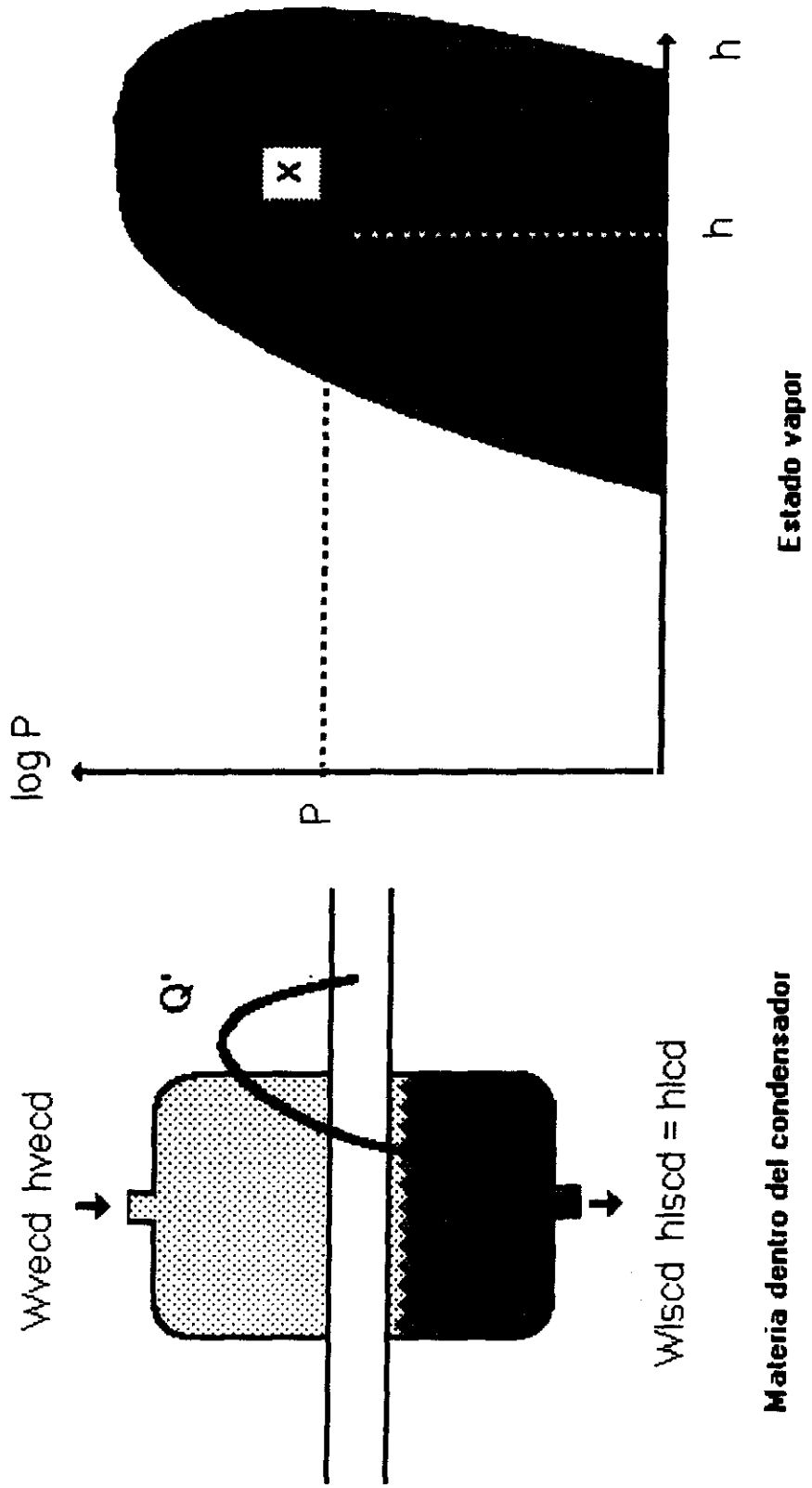

h
x
h
log P
P
Estado vapor
Q'
Wvecd hvecd
Wlscd hlscd = hlcd
Materia dentro del condensador

Figura 5.16

- Según la suposición $P_r = P_{cd}$ (ver fig. 5.17), tenemos que, <u>para una T_{rs} dada</u>:

$$h_{rs} = \Phi(P_{cd}) \rightarrow \begin{cases} h_{rs} = e_0 + e_1 P_{cd} + \ldots + e_n P_{cd}^n \\ \dfrac{dh_{rs}}{dP_{cd}} = e_1 + \ldots + n\, e_n P_{cd}^{n-1} = k_5 \\ \dfrac{dh_{rs}}{dt} = k_5 \dfrac{dP_{cd}}{dt} \end{cases}$$

- Desarrollo de las ecuaciones:

Balance de masa:

$$W_{vecd} - W_{lscd} = \frac{dm_{lcd}}{dt} + \frac{dm_{vecd}}{dt} + \frac{dm_m}{dt} =$$

$$= \frac{d(\rho_{lcd} V_{lcd})}{dt} + \frac{d(\rho_{vecd} V_{vcd})}{dt} + \frac{d(\rho_m V_{vcd})}{dt} = \rho_{lcd} \frac{dV_{lcd}}{dt} + V_{lcd} \frac{d\rho_{lcd}}{dt} + \rho_{vecd} \frac{dV_{vcd}}{dt} +$$

$$+ V_{vcd} \frac{d\rho_{vecd}}{dt} + \rho_m \frac{dV_{vcd}}{dt} + V_{vcd} \frac{d\rho_m}{dt} = \rho_{lcd}\, \Pi\, R^2 \frac{dn}{dt} + \Pi\, R^2\, n\, k_2 \frac{dP}{dt} +$$

$$+ \rho_{vecd} (-\Pi\, R^2 \frac{dn}{dt}) + \Pi\, R^2 (L - n)\, k_{11} \frac{dP}{dt} + \rho_m (-\Pi\, R^2 \frac{dn}{dt}) + \Pi\, R^2 (L - n)\, k_{12} \frac{dP}{dt} =$$

$$= \frac{dn}{dt} (\Pi\, R^2\, \rho_{lcd} - \Pi\, R^2\, \rho_{vecd} - \Pi\, R^2\, \rho_m) + \frac{dP}{dt} (\Pi\, R^2\, n\, k_2 + \Pi\, R^2 (L - n)\, k_{11} +$$

$$+ \Pi\, R^2 (L - n)\, k_{12})$$

Llamando : $\begin{cases} A_0 = \Pi\, R^2 \\ A_1 = \rho_{lcd} - \rho_{vecd} - \rho_m \\ A_2 = V_{lcd}\, k_2 + V_{vcd}\, k_{11} + V_{vcd}\, k_{12} \end{cases}$, entonces:

$$\boxed{W_{vecd} - W_{lscd} = A_0\, A_1 \frac{dn_{cd}}{dt} + A_2 \frac{dP_{cd}}{dt}}$$

(I)

Balance de energía:

$$\frac{d}{dt}(m_{lcd}\, h_{lcd} + m_{vecd}\, h_{vecd} + m_m\, h_m) = W_{vecd}\, h_{vecd} - W_{lscd}\, h_{lcd} + Q' \Rightarrow$$

$$\frac{d}{dt}(\rho_{lcd}\, V_{lcd}\, h_{lcd} + \rho_{vecd}\, V_{vcd}\, h_{vecd} + \rho_m\, V_{vcd}\, h_m) = \rho_{lcd}\, V_{lcd} \frac{dh_{lcd}}{dt} +$$

$$+ \rho_{lcd}\, h_{lcd} \frac{dV_{lcd}}{dt} + V_{lcd}\, h_{lcd} \frac{d\rho_{lcd}}{dt} + \rho_{vecd}\, V_{vcd} \frac{dh_{vecd}}{dt} + \rho_{vecd}\, h_{vecd} \frac{dV_{vcd}}{dt} +$$

$$+ V_{vcd}\, h_{vecd} \frac{d\rho_{vecd}}{dt} + \rho_m\, V_{vcd} \frac{dh_m}{dt} + \rho_m\, h_m \frac{dV_{vcd}}{dt} + V_{vcd}\, h_m \frac{d\rho_m}{dt} = \rho_{lcd}\, V_{lcd}\, k_4 \frac{dP}{dt} +$$

$$+ \rho_{lcd}\, h_{lcd}\, \Pi\, R^2 \frac{dn}{dt} + V_{lcd}\, h_{lcd}\, k_2 \frac{dP}{dt} + \rho_{vecd}\, V_{vcd}\, k_{31} \frac{dP}{dt} + \rho_{vecd}\, h_{vecd} (-\Pi\, R^2 \frac{dn}{dt}) +$$

$$+ V_{vcd}\, h_{vecd}\, k_{11} \frac{dP}{dt} + \rho_m\, V_{vcd}\, k_{32} \frac{dP}{dt} + \rho_m\, h_m (-\Pi\, R^2 \frac{dn}{dt}) + V_{vcd}\, h_m\, k_{12} \frac{dP}{dt} =$$

$$= \frac{dn}{dt}(\rho_{lcd}\, h_{lcd}\, \Pi\, R^2 - \rho_{vecd}\, h_{vecd}\, \Pi\, R^2 - \rho_m\, h_m\, \Pi\, R^2) + \frac{dP}{dt}(\rho_{lcd}\, V_{lcd}\, k_4 +$$

$$+ V_{lcd}\, h_{lcd}\, k_2 + \rho_{vecd}\, V_{vcd}\, k_{31} + V_{vcd}\, h_{vecd}\, k_{11} + \rho_m\, V_{vcd}\, k_{32} + V_{vcd}\, h_m\, k_{12}) =$$

$$= W_{vecd}\, h_{vecd} - W_{lscd}\, h_{lscd} - m_r \frac{dh_{rs}}{dt} + W_r\,(h_{re} - h_{rs})$$

Llamando:
$$\begin{cases} A_3 = h_{lcd}\, \rho_{lcd} - h_{vecd}\, \rho_{vecd} - \rho_m\, h_m \\ A_4 = k_{11}\, h_{vecd}\, V_{vcd} + k_{12}\, h_m\, V_{vcd} + k_2\, h_{lcd}\, V_{lcd} + k_{31}\, \rho_{vecd}\, V_{vcd} + \\ \qquad + k_{32}\, \rho_m\, V_{vcd} + k_4\, \rho_{lcd}\, V_{lcd} \end{cases}$$

entonces

$$A_0\, A_3 \frac{dn_{cd}}{dt} + A_4 \frac{dP_{cd}}{dt} = W_{vecd}\, h_{vecd} - W_{lscd}\, h_{lcd} - m_r \frac{dh_{rs}}{dt} + W_r\,(h_{re} - h_{rs})$$

Sabiendo que $\frac{dh_{rs}}{dt} = k_5 \frac{dP_{cd}}{dt}$, y llamando $A'_4 = A_4 + m_r k_5$:

$$A_0 A_3 \frac{dn_{cd}}{dt} + A'_4 \frac{dP_{cd}}{dt} = W_{vecd} h_{vecd} - W_{lscd} h_{lcd} + W_r (h_{re} - h_{rs})$$

(II)

Así pues, con (I) y (II) tenemos dos ecuaciones con dos incógnitas, $\frac{dn_{cd}}{dt}$ y $\frac{dP_{cd}}{dt}$; operando con ambas, podemos llegar a las siguientes expresiones fundamentales, que calculan el nivel y la presión en el condensador:

$$\frac{dn_{cd}}{dt} = \frac{(\frac{A'_4}{A_2} - h_{vecd})\, W_{vecd} + (h_{lcd} - \frac{A'_4}{A_2})\, W_{lscd} - W_r (h_{re} - h_{rs})}{\frac{A_0 A_1 A'_4}{A_2} - A_0 A_3} = A$$

$$\frac{dP_{cd}}{dt} = \frac{(\frac{A_3}{A_1} - h_{vecd})\, W_{vecd} + (h_{lcd} - \frac{A_3}{A_1})\, W_{lscd} - W_r (h_{re} - h_{rs})}{\frac{A_2 A_3}{A_1} - A'_4} = B$$

- Consideraciones finales:

Este modelo exige que nosotros debemos fijar las temperaturas de entrada y salida del refrigerante (T_{re}, T_{rs}). O sea, T_{rs} no dependerá de las demás variables. Esto es porque el modelo considera que el sistema vapor + líquido saturado + refrigerante está a una misma presión, y que el calor absorbido por el vapor + líquido saturado es el cedido por el refrigerante, calor que depende de la diferencia de estados del refrigerante a la entrada y a la salida. Otros modelos pueden considerar otras dinámicas con otras variables de estado.

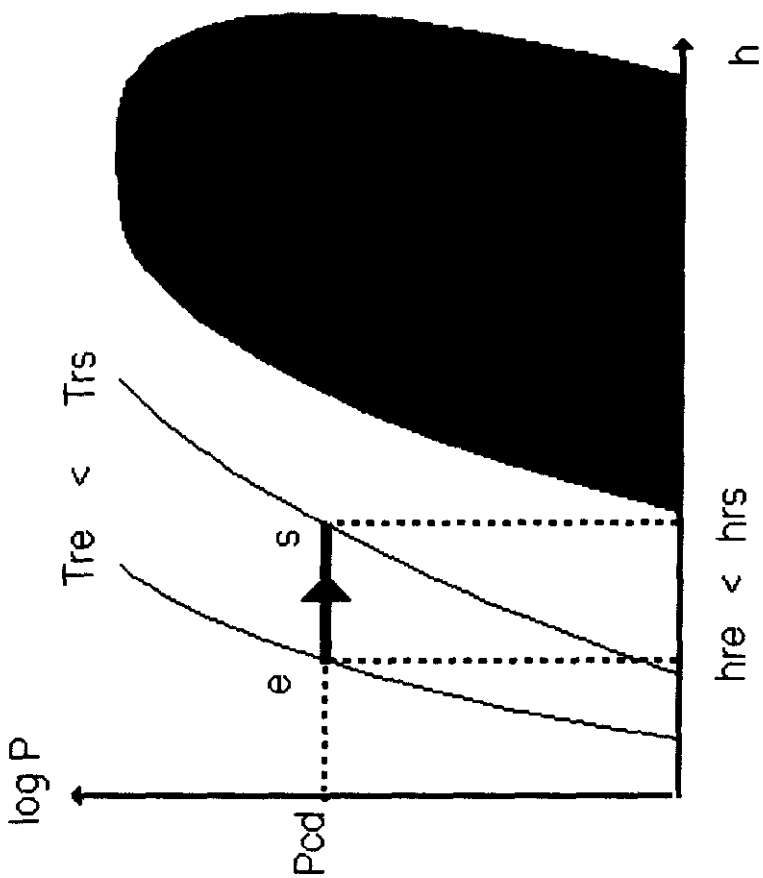


Figura 5.17

MODELO ORIENTADO A OBJETO.

En la figura 5.18 podemos ver esquematizada la implementación del anterior modelo matemático en la clase *Condensador*. Podemos destinguir los siguientes miembros:

— *Protegidos:*
Variables que acogen las dimensiones del recinto, los valores de las magnitudes termodinámicas de los elementos de entrada, salida y refrigeración, constantes, variables para la resolución de las ecuaciones, nivel, presión, etc. Un objeto de la clase *Líquido_saturado* representará al elemento de salida.

— *Públicos:*
El constructor inicializará los parámetros fundamentales. Sendas funciones asignarán los valores de cada fluido, y otra función resolverá las ecuaciones diferenciales para el cálculo del nivel y la presión.

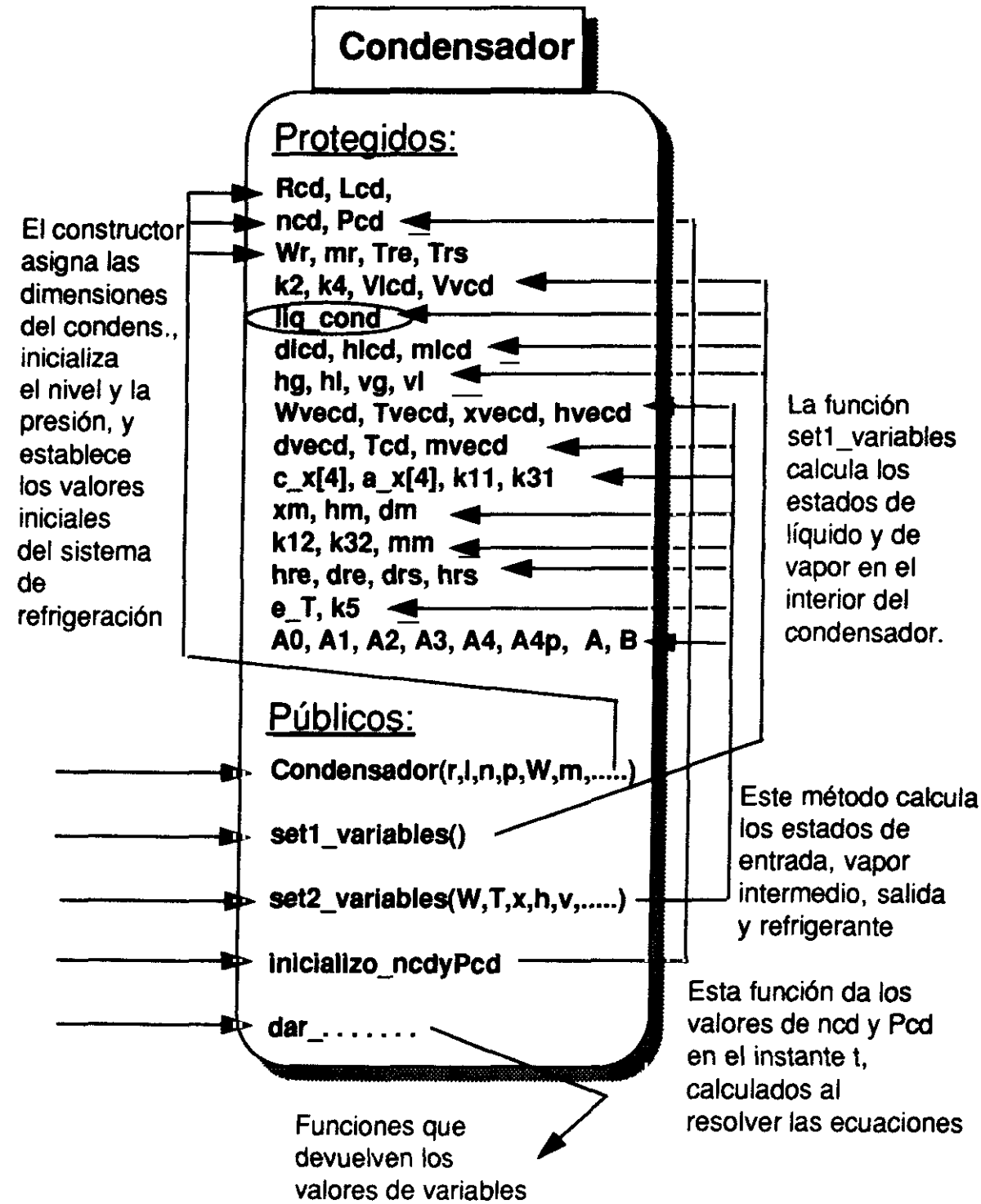

Condensador
Protegidos:
Rcd, Lcd,
ncd, Pcd
Wr, mr, Tre, Trs
k2, k4, Vlcd, Vvcd
liq_cond
dlcd, hlcd, mlcd
hg, hl, vg, vl
Wvecd, Tvecd, xvecd, hvecd
dvecd, Tcd, mvecd
c_x[4], a_x[4], k11, k31
xm, hm, dm
k12, k32, mm
hre, dre, drs, hrs
e_T, k5
A0, A1, A2, A3, A4, A4p, A, B
Públicos:
Condensador(r,l,n,p,W,m,.....)
set1_variables()
set2_variables(W,T,x,h,v,.....)
inicializo_ncdyPcd
dar_.....
El constructor asigna las dimensiones del condens., inicializa el nivel y la presión, y establece los valores iniciales del sistema de refrigeración
La función set1_variables calcula los estados de líquido y de vapor en el interior del condensador.
Este método calcula los estados de entrada, vapor intermedio, salida y refrigerante
Esta función da los valores de ncd y Pcd en el instante t, calculados al resolver las ecuaciones
Funciones que devuelven los valores de variables
= objeto

Figura 5.18

V.2.11.- BOMBA.-

MODELO TEORICO.

La bomba es un dispositivo que eleva la presión del líquido entrante. En nuestro caso, vamos a efectuar la suposición de que entra líquido saturado y sale en estado líquido (ver figura 5.19). Así pues, dentro de la bomba están todos los estados intermedios entre el líquido saturado y el estado final de líquido, que será el correspondiente a la presión de salida (ver figura 5.20). Ahora bien, esta presión de salida la marca la caldera, pues el líquido de salida de la bomba es el de entrada de la caldera, de forma que suponemos que está todo ello a la misma presión P_c (en el interior de la caldera habrá una mezcla de estados que van desde el líquido hasta el líquido saturado, si bien habíamos hecho la suposición de que el interior de la caldera se encuentra en equilibrio saturado).

Por otro lado, y debido a la difícil compresibilidad del agua en su estado líquido, y a efectos de no complicar los cálculos, vamos a suponer que la densidad (o volumen específico) de los fluidos de entrada y de salida son iguales.

Por último, consideraremos que no hay pérdida ni almacenamiento de agua en la bomba, de forma que el caudal de entrada sea igual al de salida. Por tanto, nuestros supuestos teóricos se resumen en:

$$\rho_{lsb} = \rho_{leb} \qquad P_{lsb} = P_c \qquad W_{lsb} = W_{leb} = W_b$$

Para calcular los valores de las distintas magnitudes termodinámicas, hemos de conocer la presión en la caldera, la presión y el caudal del líquido saturado de entrada.

- Conozco: P_c, P_{leb} y W_{leb}.
- Entrada: d_{leb} y T_{leb} se hallan a partir de las tablas de vapor (dato: P_{leb}).
- Salida:

$$\rho_{lsb} = \rho_{leb} \mid P_{lsb} = P_c \mid \left.\begin{matrix} P_{lsb} \\ \rho_{lsb} \end{matrix}\right\} \rightarrow \text{(mediante tablas)} \rightarrow h_{lsb}\,,\ T_{lsb}$$

MODELO ORIENTADO A OBJETO.

En la figura 5.21 vemos esquematizada la clase *Bomba*, donde el constructor establece el caudal; una función protegida calcula el estado de salida y una pública asigna la entrada. Otros métodos nos permiten controlar el valor de Wb y devolver los valores de diversas magnitudes.

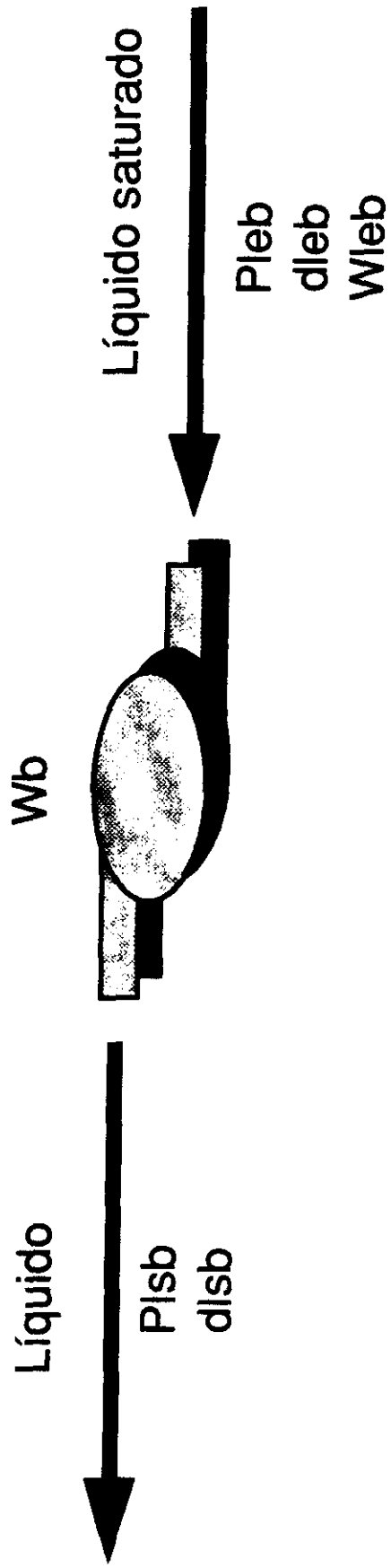


Figura 5.19

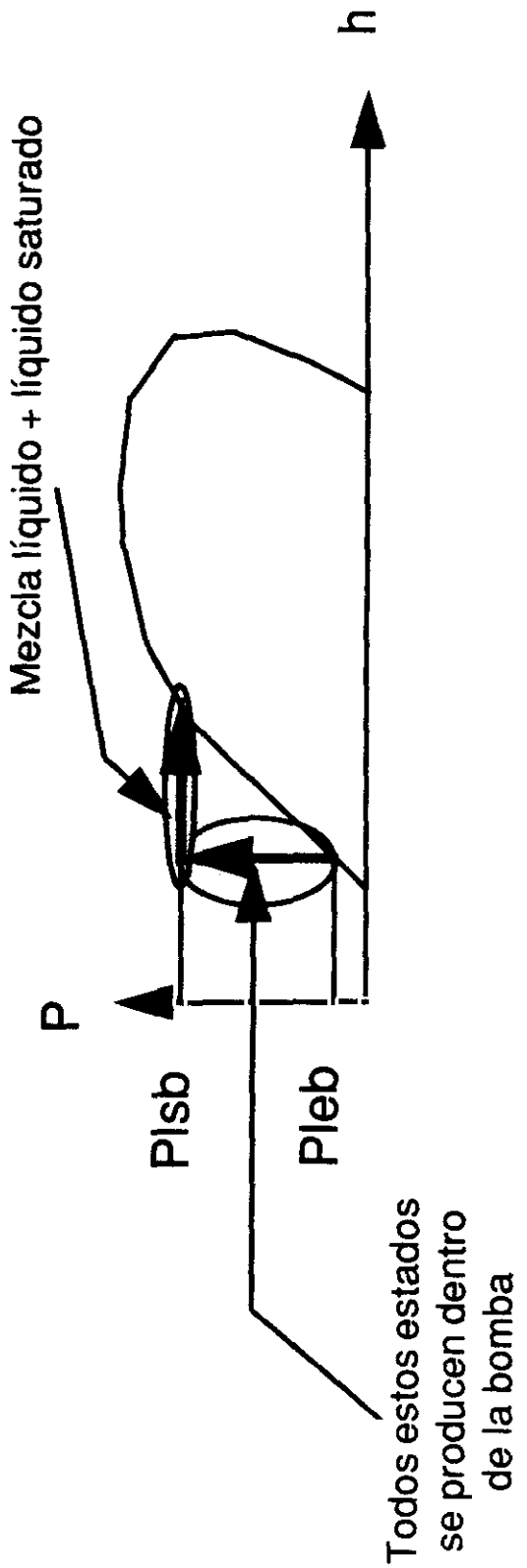

h
Mezcla líquido + líquido saturado
P
Plsb
Pleb
Todos estos estados
se producen dentro
de la bomba

Figura 5.20

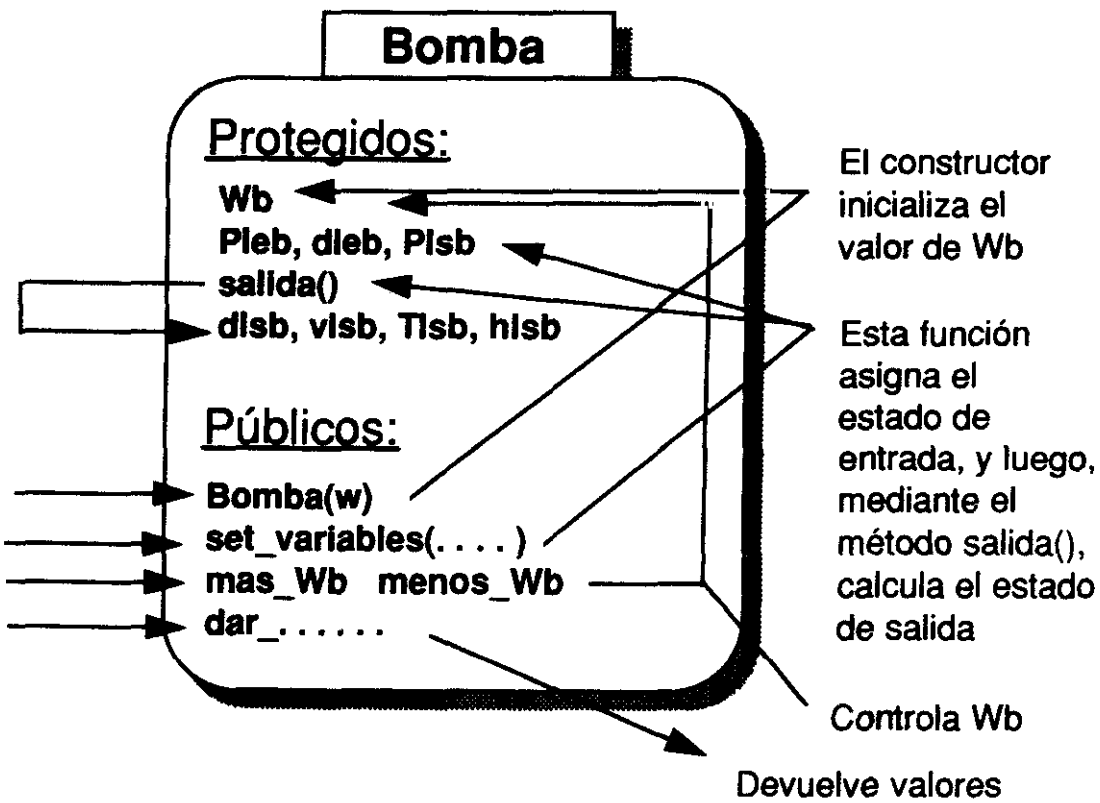

Bomba
Protegidos:
Wb
Pieb, dieb, Pisb
salida()
disb, visb, Tisb, hisb
Públicos:
Bomba(w)
set_variables(. . . .)
mas_Wb menos_Wb
dar_. . . . . .
El constructor inicializa el valor de Wb
Esta función asigna el estado de entrada, y luego, mediante el método salida(), calcula el estado de salida
Controla Wb
Devuelve valores

Figura 5.21

V.3.- MODELACION DE ESTADOS TERMODINAMICOS MEDIANTE TABLAS.-

En este apartado explicaremos cómo hemos calculado los valores de las distintas magnitudes termodinámicas a partir de las tablas de vapor, mediante la búsqueda, identificación e interpolación de los valores adecuados.

La situación de la que partimos es la siguiente: Tenemos conocimiento del valor de uno o más parámetros de cierto estado, y queremos, a partir de esto, averiguar los restantes valores con el fin de completar la descripción del estado. Para ello, acudimos a las tablas disponibles en las referencias bibliográficas. La primera dificultad con que nos encontramos es tener que incluír dichas tablas en nuestros programas, puesto que no existen ecuaciones que las modelen adecuadamente. Se impone, pues, efectuar dos tareas:

- 1) Incorporación de las tablas de vapor a nuestros programas, mediante arrays de datos de una y dos dimensiones. Esto nos obliga a una tediosa tarea de escritura.

- 2) Elaboración de los algoritmos de gestión de estos arrays (tablas) que identifiquen, busquen e interpolen valores.

Para la segunda tarea, hemos de tener en cuenta que pueden darse dos casos: tabla unidimensional ó bidimensional. En el primer caso, conociendo un valor (v.g. x[35]), obtenemos otro (v.g. y[35]) o más del mismo estado, conociendo tan sólo el índice (35) correspondiente; en el segundo caso, necesito dos valores de entrada (v.g x[12] e y[43]) para obtener un tercer valor (z[12][43]) o más del mismo estado. En este último caso, no siempre el dato o los datos conocidos coinciden con las variables que dimensionan la tabla (x e y), sino con el resultado de la misma (z), de forma que es necesario reorganizar la tabla, lo que equivale a crear otra nueva, o bien a ingeniar algoritmos que nos saquen del apuro.

Cada tabla da la solución de una magnitud termodinámica (temperatura, presión, volumen específico, entropía y entalpía) a partir del conocimiento de otra u otras. Por otra parte, cada tabla hace referencia a uno de los estados concretos del agua (líquido, líquido saturado, vapor, vapor saturado y vapor supercalentado).

En este gráfico Presión-Entalpia podemos ver distintas magnitudes termodinámicas, y distintas formas de buscarlas según el estado en el que nos encontremos. Debido a la inexactitud con que cabe obtener valores de esta gráfica, además de la dificultad de tratarla con un programa informático, acudimos a las tablas. A su vez, éstas han de ser utilizadas de una forma conveniente para lograr la mayor exactitud posible en nuestras interpolaciones.

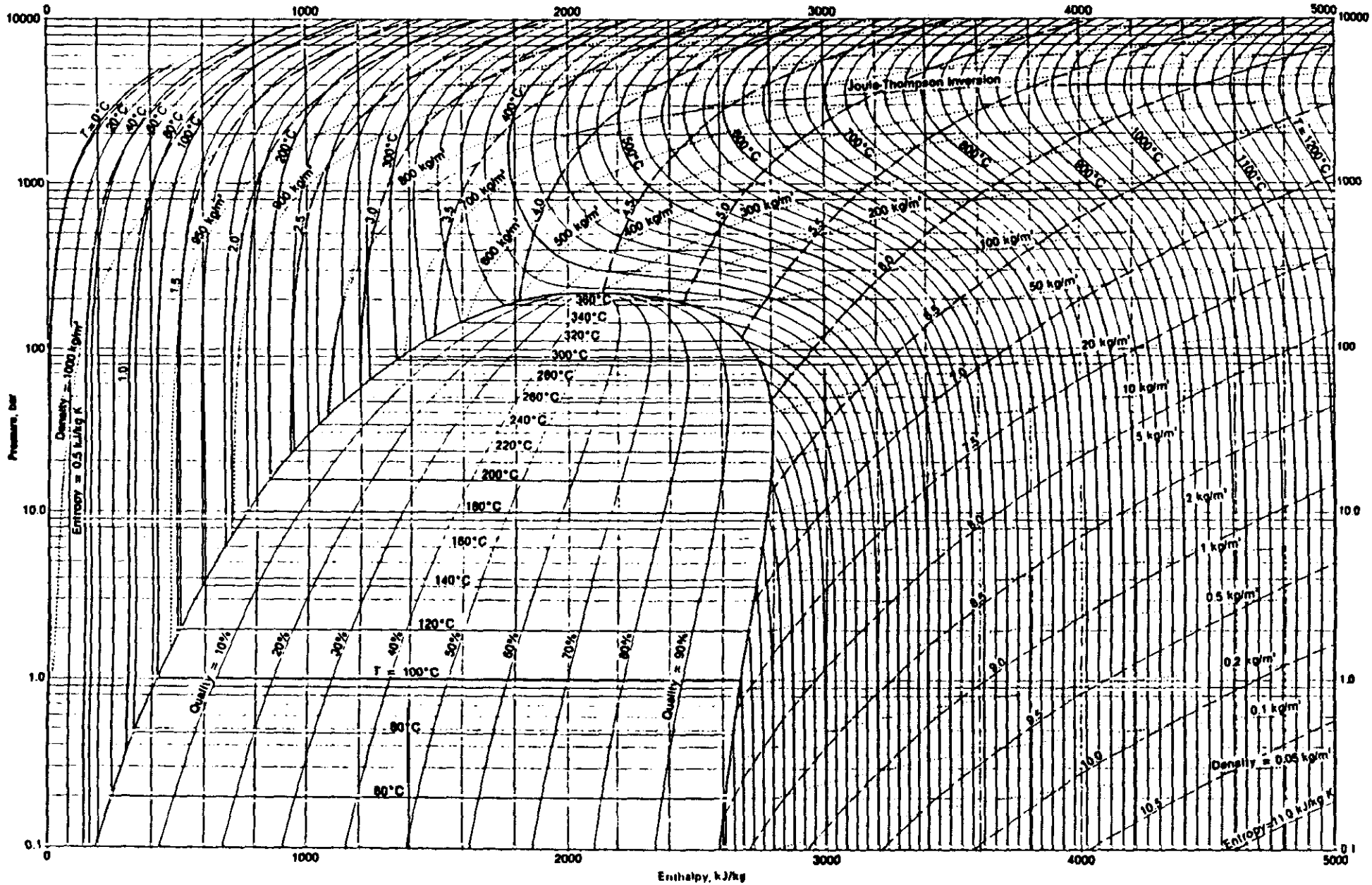


Figure D.2 Pressure-enthalpy diagram for steam. [From Lester Haar, John S. Gallagher, and George S. Kell, NBS/NRC Steam Tables, 1984. With permission from Hemisphere Publishing Corporation, New York.]

A continuación, mostramos los arrays empleados : sus descriptores, magnitudes de dimensión (dos, en caso de tabla bidimensional), magnitud que devuelve (con sus unidades) y estado al que pertenece. Pueden verse en los listados que incluímos en los apéndices.

DESCRIPTOR	Magnitud x	Magnitud y	MAGNITUD	ESTADO
T_sat[76]			Temperatura (ºC)	Vapor saturado
P_sat[76]			Presión (KPa)	Vapor saturado
h_sat_v[76]			Entalpía (KJ/Kg)	Vapor saturado
s_sat_v[76]			Entropía (KJ/(Kg.ºK))	Vapor saturado
v_sat_v[76]			Volumen específico (m3/Kg)	Vapor saturado
P_l[27]			Presión (KPa)	Líquido
T_l[27][22]	Presión	Temperatura	Temperatura (ºC)	Líquido
h_l[27][22]	Presión	Temperatura	Entalpía (KJ/Kg)	Líquido
v_l[27][22]	Presión	Temperatura	Volumen específico (m3/Kg)	Líquido
h_sat_l[76]			Entalpía (KJ/Kg)	Líquido saturado
s_sat_l[76]			Entropía (KJ/(Kg.ºK))	Líquido saturado
v_sat_l[76]			Volumen específico (m3/Kg)	Líquido saturado
P_sup[28]			Presión (KPa)	Vapor supercalentado
TK_sup[28][18]	Presión	Temperatura	Temperatura (ºK)	Vapor supercalentado
h_sup[28][18]	Presión	Temperatura	Entalpía (KJ/Kg)	Vapor supercalentado
s_sup[28][18]	Presión	Temperatura	Entropía (KJ/(Kg.ºK))	Vapor supercalentado
v_sup[28][18]	Presión	Temperatura	Volumen específico (m3/Kg)	Vapor supercalentado

El estado de vapor está descrito no por tablas, sino por ecuaciones de interpolación que serán convenientemente explicadas en V.3.3.

Estas tablas son gestionadas por unas funciones que serán descritas más adelante. La peculiar disposición de las tablas de vapor en arrays, se debe a la mayor facilidad de acceder a los procesos de búsqueda e interpolación de valores.

V.3.1.- GESTION DE TABLAS GENERICAS.-

a) TABLA UNIDIMENSIONAL.

La tabla es de de esta forma:

x[size]	y[size]	**Indice**
~	~	i=0
~	~	i=1
~	~	
x_i	y_i	
x_{i+1}	y_{i+1}	
~	~	
~	~	
~	~	
~	~	i=size

Tenemos dos arrays de datos, de tamaño *size*, de forma que, conocido un x_0, queremos hallar el correspondiente valor y_0. A continuación mostramos el algoritmo usado en el código C. El diagrama de flujo puede verse en la figura 5.22.

```
double tabla_1d ( double x0, double y[], double x[], int size )
        {
        double y0;
        int i;
        for( i=0; i<=size-1; i++ )
                if( x0 == x[i] )
                        { y0=y[i]; i=size; }
                else if ( x0>x[i] && x0<x[i+1] )
                        { y0=y[i]+(x0-x[i])*(y[i+1]-y[i])/(x[i+1]-x[i]); i=size; }
        return(y0);
        }
```

b) TABLA BIDIMENSIONAL.

Disponemos de una tabla de dos dimensiones, cuyos datos guardaremos en arrays de la siguiente forma:

— Dos arrays unidimensionales x[xsize] e y[ysize], de dimensiones respectivas **xsize** e **ysize**, y que guardarán los datos de las magnitudes **x** e **y**.

— Los datos de la magnitud **z** se guardarán en el array bidimensional z[xsize][ysize].

Suponemos que para cada par de datos x[i] e y[j] de la tabla, existe un dato z[i][j] también tabulado (más adelante estudiaremos el problema que surge cuando en la tabla hay pares de datos x,y que no encuentran el correspondiente z tabulado).

	y	j=0	j=1	j=2				j=ysize-1
x		y_0	y_1	y_2				
i=0	x_0	z_{00}	z_{01}	z_{02}				
i=1	x_1	z_{10}	z_{11}					
i=2	x_2	z_{20}						
					z_{ij}			
i=xsize-1								

Los arrays son declarados en un fichero cabecera, y definidos en un fichero de datos.

Función intrplcn.

Esta función básica será usada por los algoritmos de gestión de las tablas bidimensionales, por lo cual la describimos ahora. Realiza una interpolación lineal entre dos valores de una tabla unidimensional.

```
double intrplcn ( double x1, double x2, double y1, double y2, double x0 )
        { double res_intrplcn = y1 + ( x0 - x1 ) * ( y2 - y1 ) / ( x2 - x1 ) ;
          return ( res_intrplcn ); }
```

Función tabla_2dxy.

Queremos construir una función que, para x0 e y0, devuelva el correspondiente z0. La función estará así declarada:

```
double tabla_2dxy ( double x[], int xsize, double y[], int ysize, double z[], double x0, double y0 )
```

donde x[], y[] y z[] son los arrays que tomarán los datos que se les pasen en la llamada a esta función, xsize e ysize son las dimensiones, y x0 e y0 los datos conocidos. Así, por ejemplo, una línea del programa puede tener la siguiente llamada :

```
h0 = tabla_2dxy ( P, 20, T, 15, h, 1345.34, 325.6 );
```

donde P es un array de presiones, T de temperaturas, h de entalpías (bidimensional), 1345.34 es un valor de presión (que no tiene porqué ser igual a uno de los valores de la tabla de presiones), y 325.6 una temperatura. La función, con todos esos datos, busca, interpola y devuelve el valor ajustado de la correspondiente entalpía, que se asignará a h0.

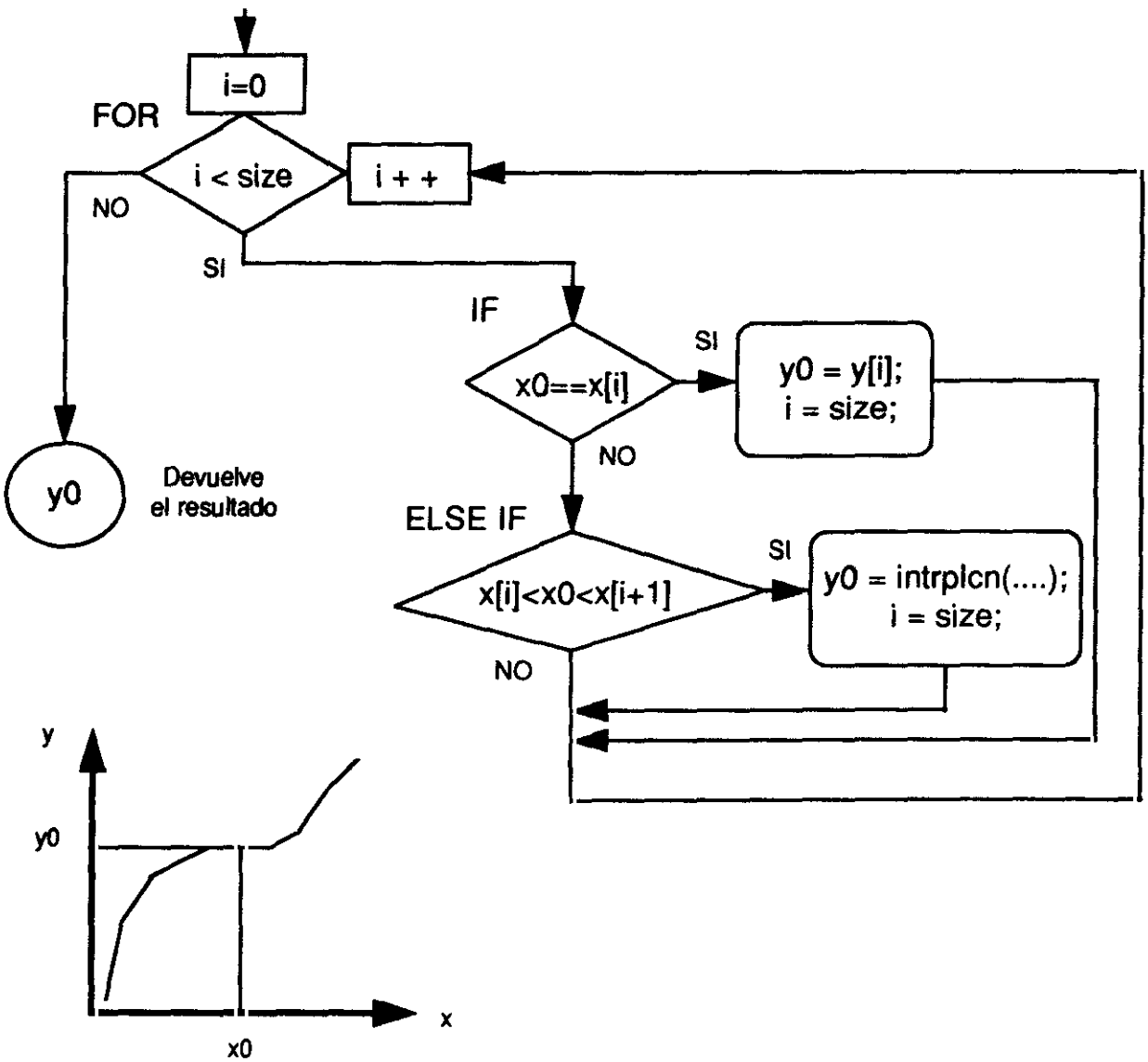

i=0
FOR
i < size
i + +
NO
SI
IF
x0==x[i]
SI
y0 = y[i];
i = size;
NO
y0
Devuelve
el resultado
ELSE IF
x[i]<x0<x[i+1]
SI
y0 = intrplcn(....);
i = size;
NO
y
y0
x
x0

Figura 5.22

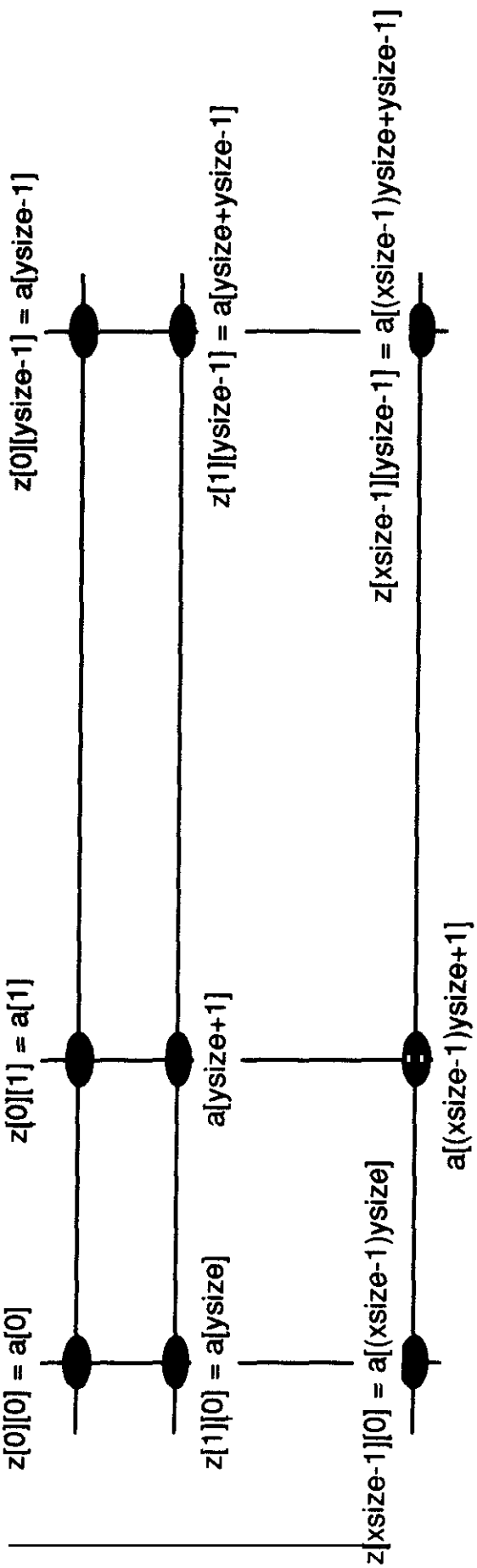

z[0][ysize-1] = a[ysize-1]
z[1][ysize-1] = a[ysize+ysize-1]
z[xsize-1][ysize-1] = a[(xsize-1)ysize+ysize-1]
z[0][1] = a[1]
a[ysize+1]
a[(xsize-1)ysize+1]
z[0][0] = a[0]
z[1][0] = a[ysize]
z[xsize-1][0] = a[(xsize-1)ysize]

Figura 5.23

Para la elaboración de nuestro algoritmo, es necesario pasar las tablas de datos x[], y[], z[][] a nuestra función tabla_2dxy(). Dentro de la función, vamos a convertir el array de dos dimensiones z[][] que le pasamos, a un array de una dimensión z[]. Utizamos una estrategia general de pasar arrays de dos dimensiones , convirtiendo en arrays de una dimensión; por ejemplo, si en el programa hacemos :

```
. . . . . .
double *a;
a = &z[0][0];
. . . . . .
```

entonces tenemos que:

```
a = z[0][0] = a[0]
a+1 = &z[0][1]  ->  *(a+1) = z[0][1] = a[1]
. . . . . .
```

como se puede ver en la figura 5.23.

Esto lo aprovecharemos para nuestra función, cuyo desarrollo podemos ver en la figura 5.24, junto con un descriptor de la estructura de arrays usada en donde puede observarse cómo hemos entendido las interpolaciones dependiendo de la posición de los valores dados. En los apéndices pueden encontrarse los listados de todas estas funciones.

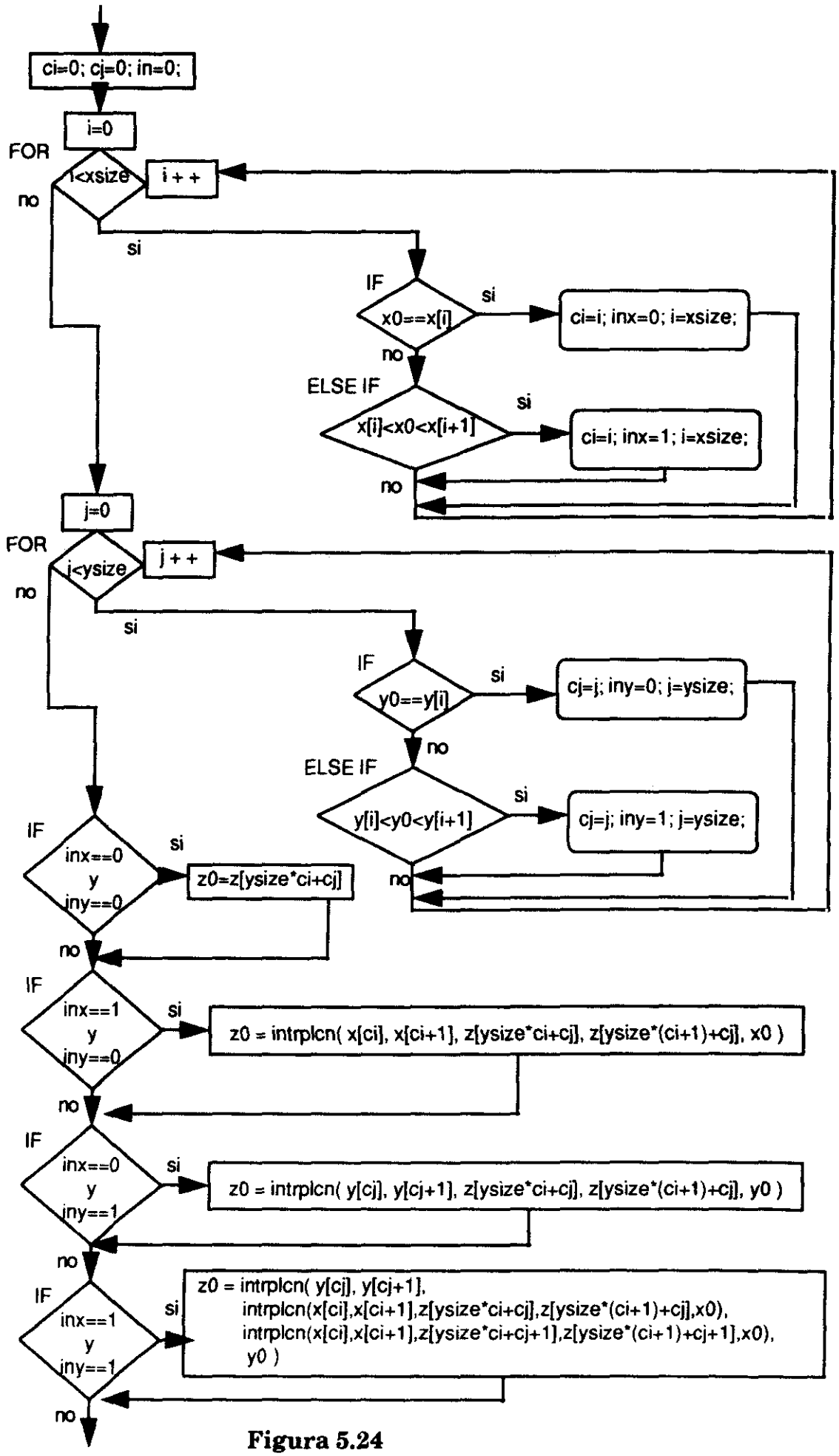

ci=0; cj=0; in=0;
i=0
FOR
i<xsize
i + +
no
si
IF
x0==x[i]
si
ci=i; inx=0; i=xsize;
no
ELSE IF
x[i]<x0<x[i+1]
si
ci=i; inx=1; i=xsize;
no
j=0
FOR
j<ysize
j + +
no
si
IF
y0==y[i]
si
cj=j; iny=0; j=ysize;
no
ELSE IF
y[i]<y0<y[i+1]
si
cj=j; iny=1; j=ysize;
no
IF
inx==0
y
iny==0
si
z0=z[ysize*ci+cj]
no
IF
inx==1
y
iny==0
si
z0 = intrplcn(x[ci], x[ci+1], z[ysize*ci+cj], z[ysize*(ci+1)+cj], x0)
no
IF
inx==0
y
iny==1
si
z0 = intrplcn(y[cj], y[cj+1], z[ysize*ci+cj], z[ysize*(ci+1)+cj], y0)
no
IF
inx==1
y
iny==1
si
z0 = intrplcn(y[cj], y[cj+1],
intrplcn(x[ci],x[ci+1],z[ysize*ci+cj],z[ysize*(ci+1)+cj],x0),
intrplcn(x[ci],x[ci+1],z[ysize*ci+cj+1],z[ysize*(ci+1)+cj+1],x0),
y0)
no

Figura 5.24

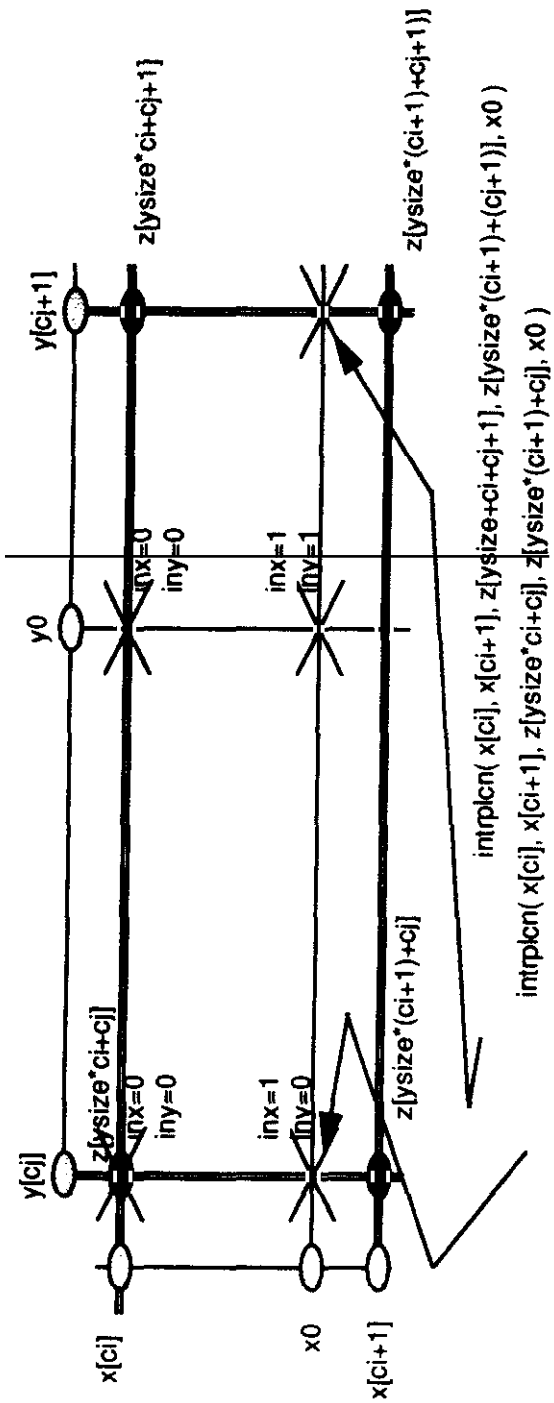

z[ysize*ci+cj+1]
z[ysize*(ci+1)+cj+1]
y[cj+1]
inx=0
iny=0
inx=1
iny=1
y0
intrplcn(x[ci], x[ci+1], z[ysize+ci+cj+1], z[ysize*(ci+1)+(cj+1)], x0)
intrplcn(x[ci], x[ci+1], z[ysize*ci+cj], z[ysize*(ci+1)+cj], x0)
z[ysize*ci+cj]
inx=0
iny=0
inx=1
iny=0
z[ysize*(ci+1)+cj]
y[cj]
x[ci]
x0
x[ci+1]

Figura 5.25

Función **tabla_2dxz.**

Queremos construir una función que, para x0 y z0 conocidos, devuelva el correspondiente valor y0. La declaración de esta función es la siguiente:

```
double tabla_2dxy ( double x[], int xsize, double y[], int ysize, double z[], double x0, double y0 )
```

Tenemos siguientes arrays:

$$\begin{cases} \text{x[xsize]} & \rightarrow \text{ por ejemplo, P[28]} \\ \text{y[ysize]} & \rightarrow \text{ por ejemplo, T[10]} \\ \text{z[xsize][ysize]} & \rightarrow \text{ por ejemplo, h[28][10]} \end{cases}$$

Entonces, en nuestro programa hacemos algo parecido a lo siguiente para llamar a nuestra función:

```
main()
        {
        double P0, T0, h0, *arr;
        arr = &z[0][0];
        P0 = . . .;
        h0 = . . .;
        T0 = tabla_2dxz( P, 28, T, 10, arr, P0, h0 );
        . . . . . .
        }
```

En la figura 5.26 podemos observar el organigrama que explica el procedimiento de búsqueda e interpolación de valores para la resolución de este caso. La definición de la función puede encontrarse en los listados que aparecen en los apéndices.

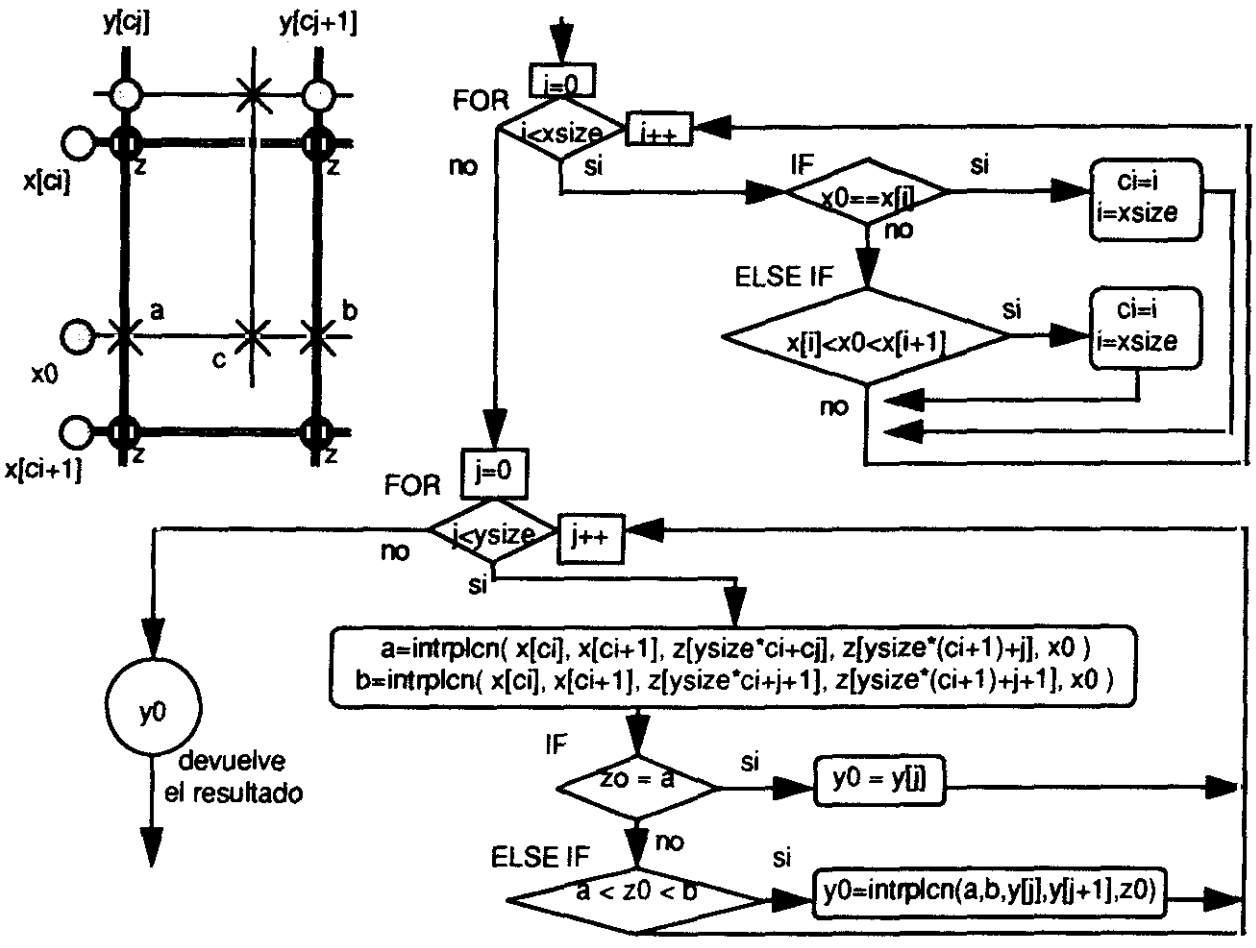

y[cj]
y[cj+1]
x[ci]
z
z
a
b
c
x0
x[ci+1]
z
z
FOR
i=0
i<xsize
i++
no
si
IF
x0==x[i]
si
ci=i
i=xsize
no
ELSE IF
x[i]<x0<x[i+1]
si
ci=i
i=xsize
no
FOR
j=0
j<ysize
j++
no
si
a=intrplcn(x[ci], x[ci+1], z[ysize*ci+cj], z[ysize*(ci+1)+j], x0)
b=intrplcn(x[ci], x[ci+1], z[ysize*ci+j+1], z[ysize*(ci+1)+j+1], x0)
IF
z0 = a
si
y0 = y[j]
ELSE IF
no
a < z0 < b
si
y0=intrplcn(a,b,y[j],y[j+1],z0)
y0
devuelve
el resultado

Figura 5.26

Función **tabla2dxzP.**

Las tablas de vapor con las que hemos contado han sido, a veces, incompletas, o bien tan insuficientes para la exactitud de cálculo deseada, que hemos tenido que completar valores a una tabla mediante valores de otra tabla. Como resultado nos encontramos con tablas bidimensionales a las que les "faltan casillas" (en cuanto a que, para un par de valores x e y no existe constancia del correspondiente z), y a las que les "sobran" (pues tenemos datos de z que no tienen correspondencia con alguna y).

Todo esto puede verse representado en la figura 5.27. Ayudados de las figuras 5.28 y 5.29 podremos seguir el organigrama de la función en las figuras 5.30 y 5.31, para el ejemplo concreto propuesto. Así, declarando la función como:

```
double tabla_2dxzP(double x[], int xsize, double y[], int ysize, double z[], double x0, double z0);
```

entonces devolverá el valor de un T para un par (P, h) dado, mediante la llamada a la función de la siguiente forma:

```
main()
        {
        double P0, h0, T0, *arrT, *arrh;
        arrh = &h[0][0];
        arrT = &T[0][0];
        P0 = . . . . ;
        h0 = . . . . ;
        T0 = tabla_2dxzP ( P, 5, arrT, 7, arrh, P0, h0 );
        . . . . . .
        }
```

La función puede verse definida en los listados correspondientes que vienen en los apéndices.

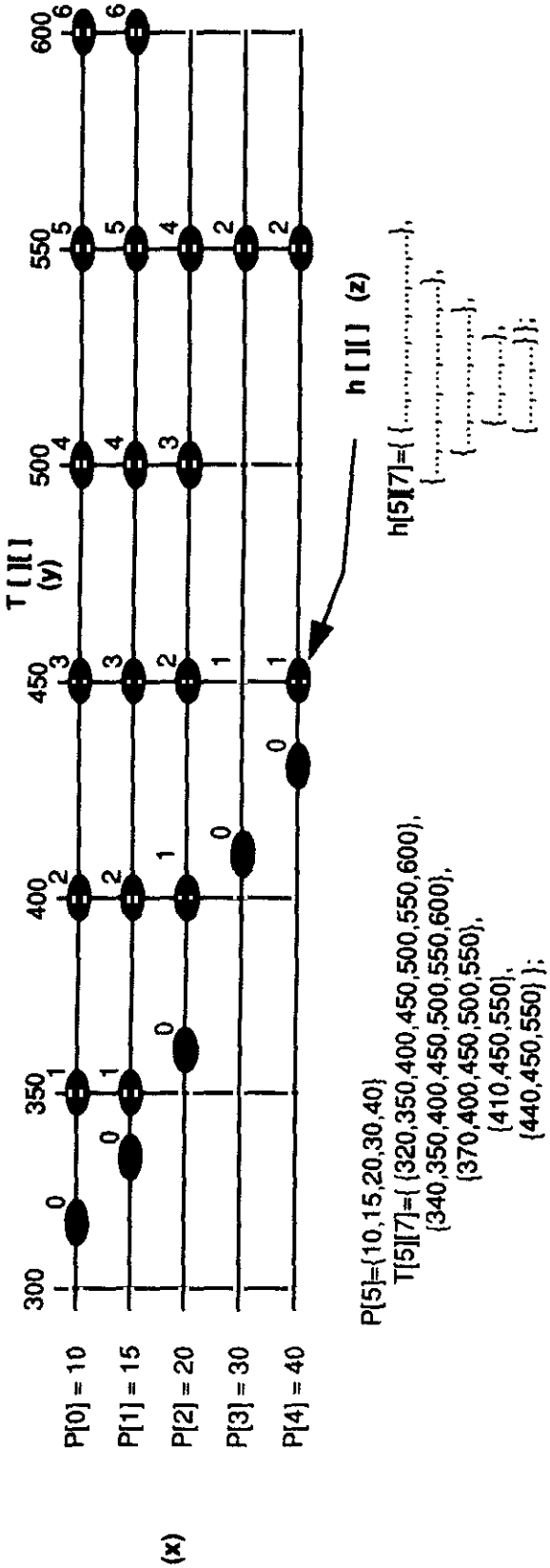

T [ ][ ]
(y)
h [ ][ ] (z)
(x)
300
350
400
450
500
550
600
P[0] = 10
P[1] = 15
P[2] = 20
P[3] = 30
P[4] = 40
P[5]={10,15,20,30,40}
T[5][7]={{320,350,400,450,500,550,600},
{340,350,400,450,500,550,600},
{370,400,450,500,550},
{410,450,550},
{440,450,550} };
h[5][7]={{.....},
{.....},
{.....},
{.....},
{.....}};

Figura 5.27

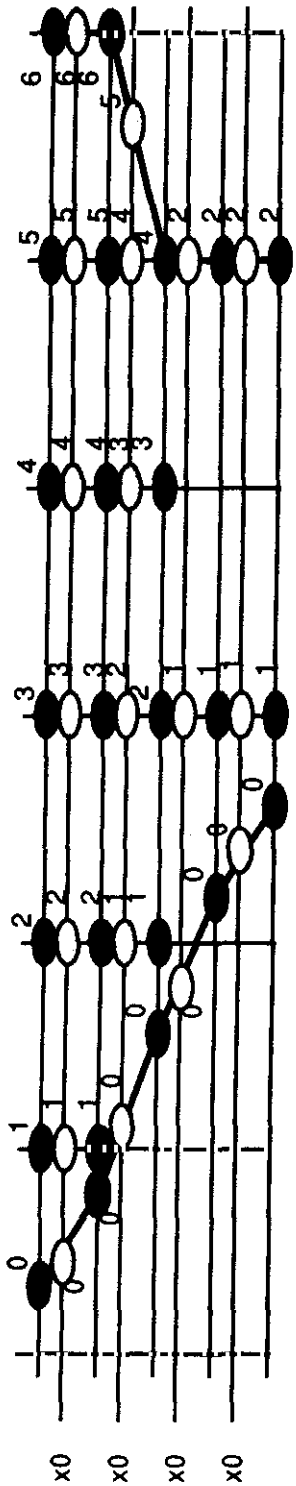


Figura 5.28

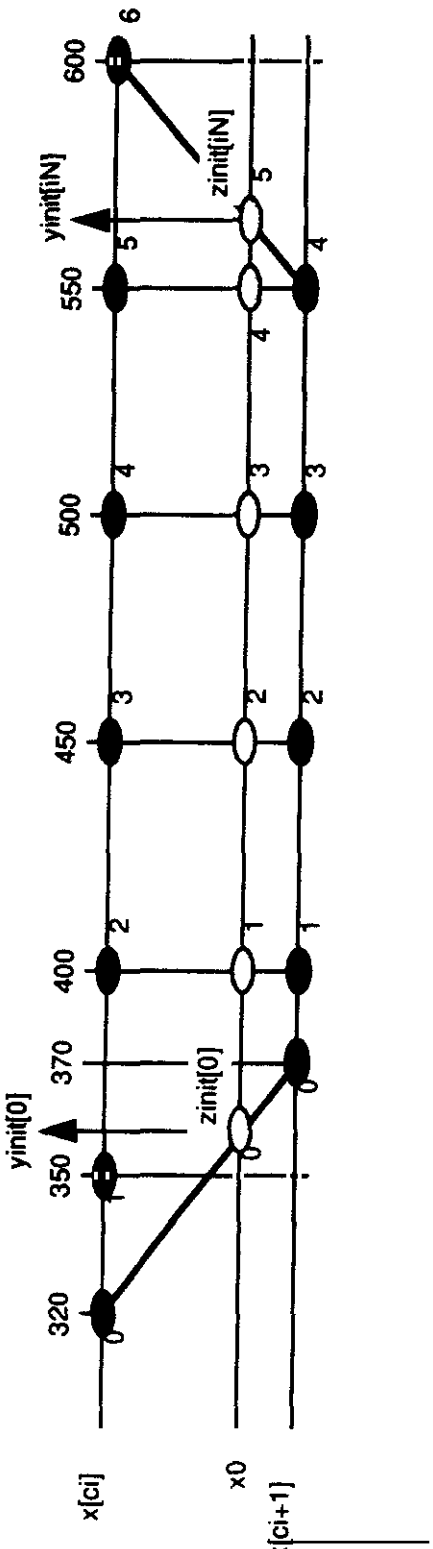

600
550
500
450
400
370
350
320
yinit[iN]
zinit[iN]
yinit[0]
zinit[0]
x[ci]
x0
x[ci+1]

Figura 5.29

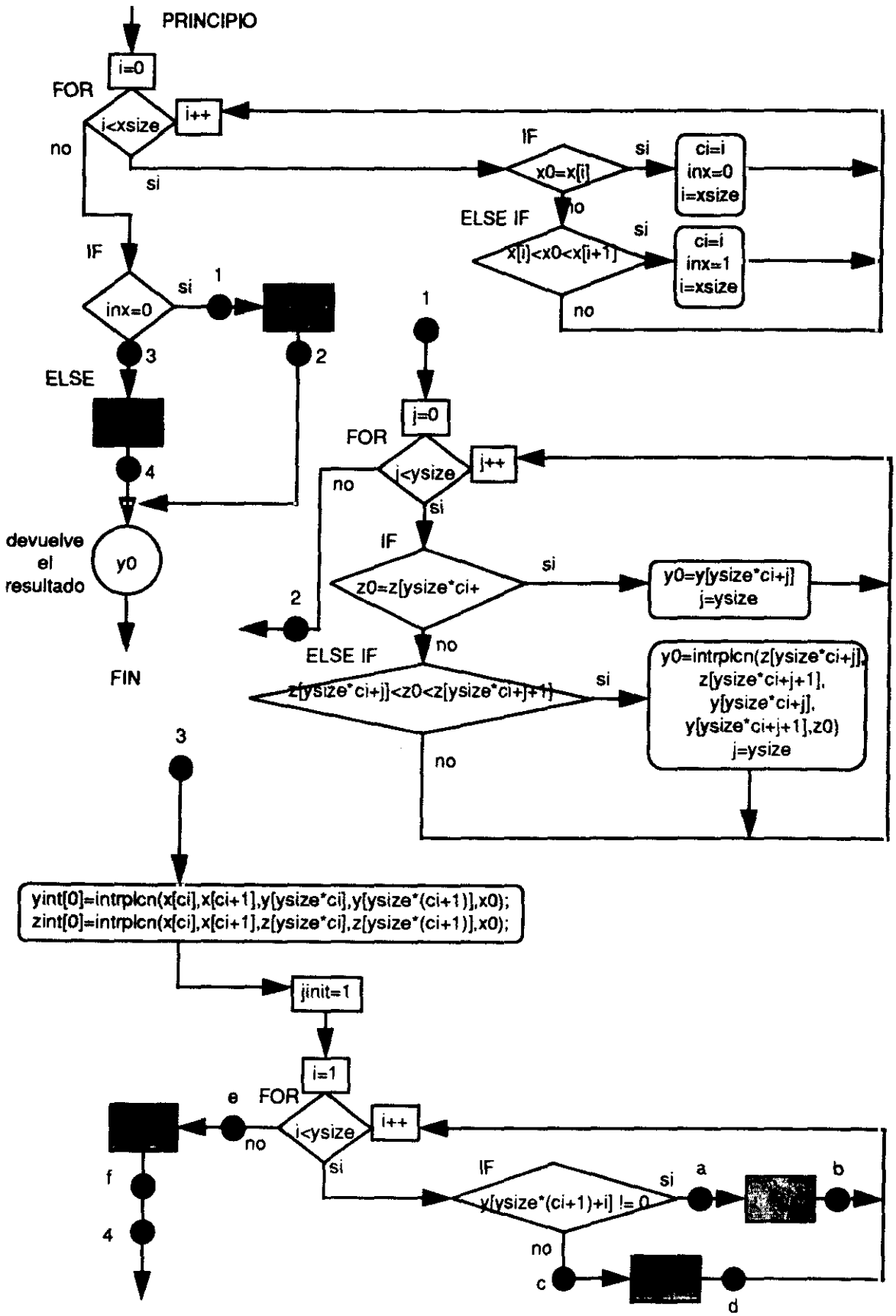

PRINCIPIO
i=0
FOR
i<xsize
i++
no
si
IF
x0=x[i]
si
ci=i
inx=0
i=xsize
ELSE IF
no
x[i]<x0<x[i+1]
si
ci=i
inx=1
i=xsize
no
IF
inx=0
si
1
3
2
ELSE
4
devuelve
el
resultado
y0
FIN
1
j=0
FOR
j<ysize
j++
no
si
IF
z0=z[ysize*ci+
si
y0=y[ysize*ci+j]
j=ysize
2
ELSE IF
no
z[ysize*ci+j]<z0<z[ysize*ci+j+1]
si
y0=intrplcn(z[ysize*ci+j],
z[ysize*ci+j+1],
y[ysize*ci+j],
y[ysize*ci+j+1],z0)
j=ysize
no
3
yint[0]=intrplcn(x[ci],x[ci+1],y[ysize*ci],y[ysize*(ci+1)],x0);
zint[0]=intrplcn(x[ci],x[ci+1],z[ysize*ci],z[ysize*(ci+1)],x0);
jinit=1
i=1
e
FOR
i<ysize
i++
no
si
f
4
IF
y[ysize*(ci+1)+i] != 0
si
a
b
no
c
d

Figura 5.30

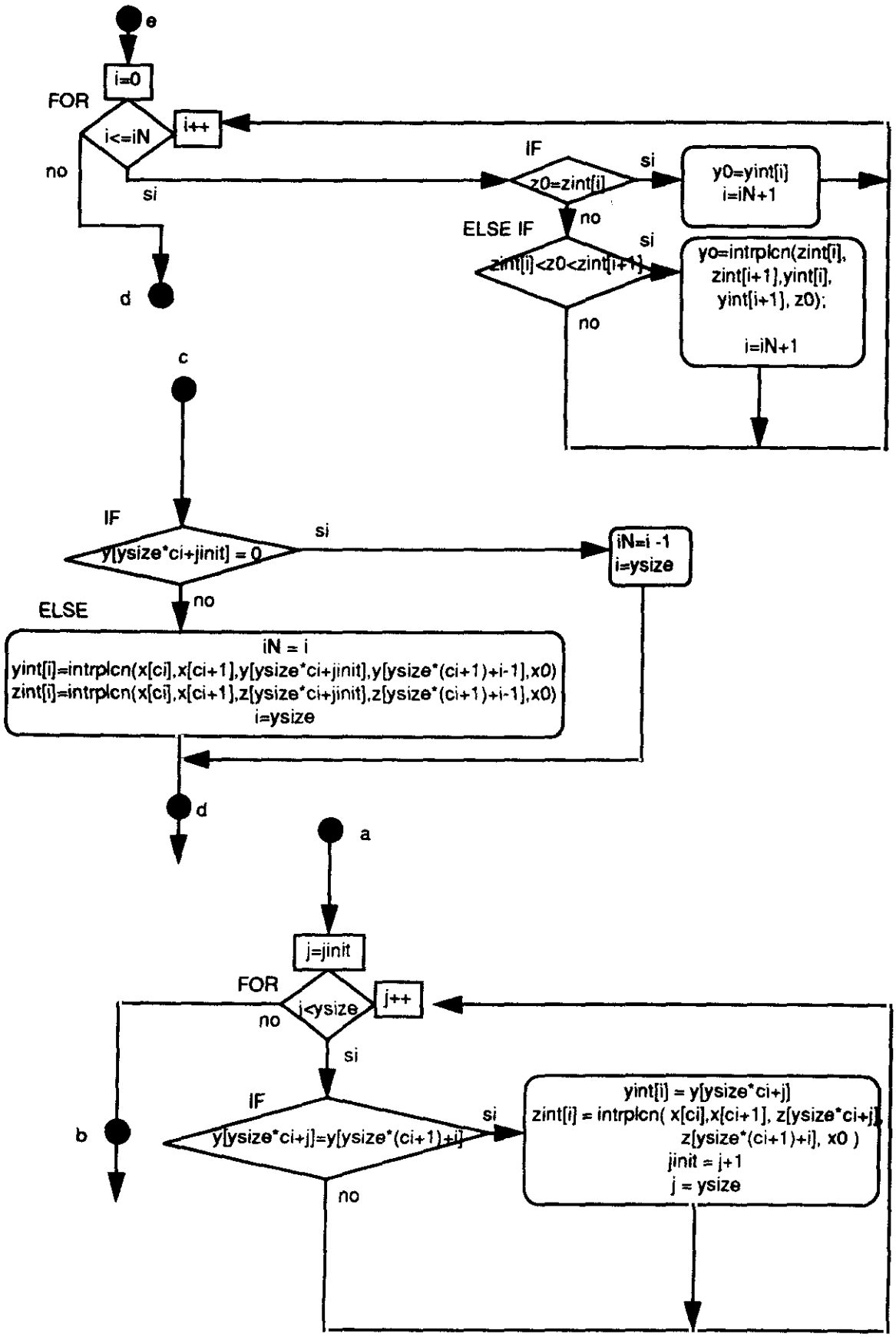

e
i=0
FOR
i<=iN
i++
no
si
d
IF
z0=zint[i]
si
y0=yint[i]
i=iN+1
no
ELSE IF
zint[i]<z0<zint[i+1]
si
yo=intrplcn(zint[i],
zint[i+1],yint[i],
yint[i+1], z0);
i=iN+1
no
c
IF
y[ysize*ci+jinit] = 0
si
iN=i -1
i=ysize
no
ELSE
iN = i
yint[i]=intrplcn(x[ci],x[ci+1],y[ysize*ci+jinit],y[ysize*(ci+1)+i-1],x0)
zint[i]=intrplcn(x[ci],x[ci+1],z[ysize*ci+jinit],z[ysize*(ci+1)+i-1],x0)
i=ysize
d
a
j=jinit
FOR
j<ysize
j++
no
si
IF
y[ysize*ci+j]=y[ysize*(ci+1)+i]
si
yint[i] = y[ysize*ci+j]
zint[i] = intrplcn(x[ci],x[ci+1], z[ysize*ci+j],
z[ysize*(ci+1)+i], x0)
jinit = j+1
j = ysize
b
no

Figura 5.31

V.3.2.- GESTION DE LAS TABLAS DEL ESTADO LIQUIDO.-

Mediante las funciones anteriores hemos podido gestionar prácticamente todas las tablas termodinámicas. Sin embargo, para la tabla del estado líquido, hemos creado dos funciones específicas: tabla_2d_xy_liq(. . .) y tabla_2d_xz_liq(. . .).

En la figura 5.32 podemos ver en un ejemplo la disposición típica de la tabla del agua en estado líquido. A partir de ella comprenderemos la forma de crear los algoritmos de búsqueda e interpolación. Las funciones creadas están descritas en las figuras 5.33 y 5.34 (tabla_2d_xy_liq) y en la 5.35 y 5.36 (tabla_2d_xz_liq), y sus códigos en lenguaje C pueden verse en los listados que vienen en los apéndices.

Las declaraciones de estas funciones son como sigue :

```
double tabla_2d_xy_liq(double x[], int xsize, double y[], int ysize, double z[], double x0, double y0);

double tabla_2d_xz_liq(double x[], int xsize, double y[], int ysize, double z[], double x0, double z0);
```

La forma de usar estas funciones sería, por ejemplo :

```
main()
        {
        double P0, T0, h0, *arrh, *arrT;
        arrh = &h[0][0];  arrT = &T[0][0];
        P0 = . . . . . ;  T0 = . . . . . ;
        h0 = tabla_2d_xy_liq( P, 4, arrT, 5, arrh, P0, T0 );
        . . . . .
        }
```

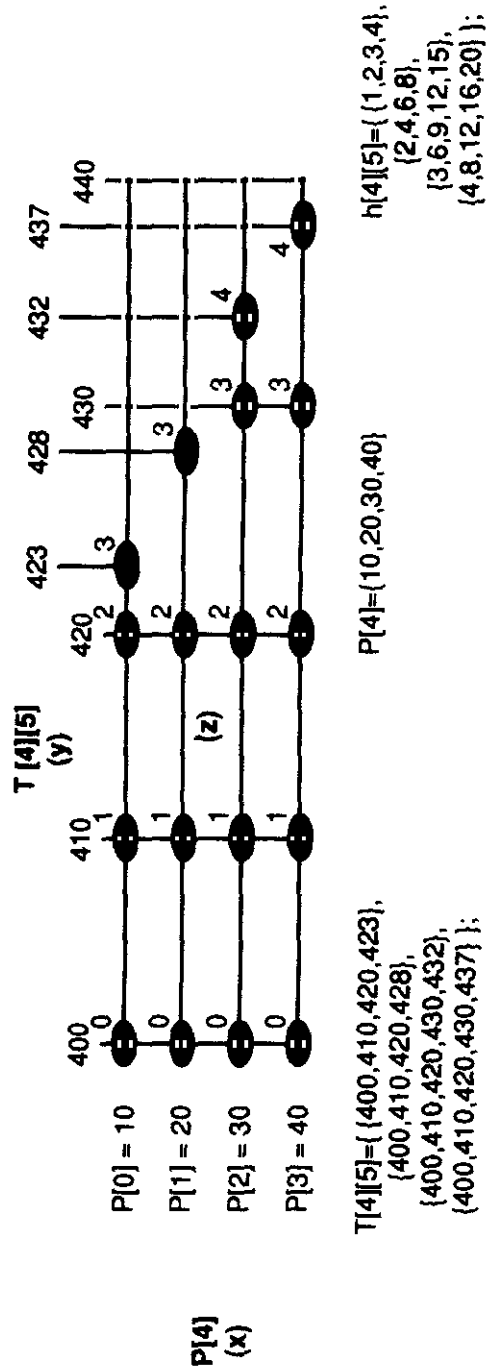

T [4][5]
(y)
P[4]
(x)
(z)
400
410
420
423
428
430
432
437
440
P[0] = 10
P[1] = 20
P[2] = 30
P[3] = 40
P[4]={10,20,30,40}
T[4][5]={ {400,410,420,423},
{400,410,420,428},
{400,410,420,430,432},
{400,410,420,430,437} };
h[4][5]={ {1,2,3,4},
{2,4,6,8},
{3,6,9,12,15},
{4,8,12,16,20} };

Figura 5.32

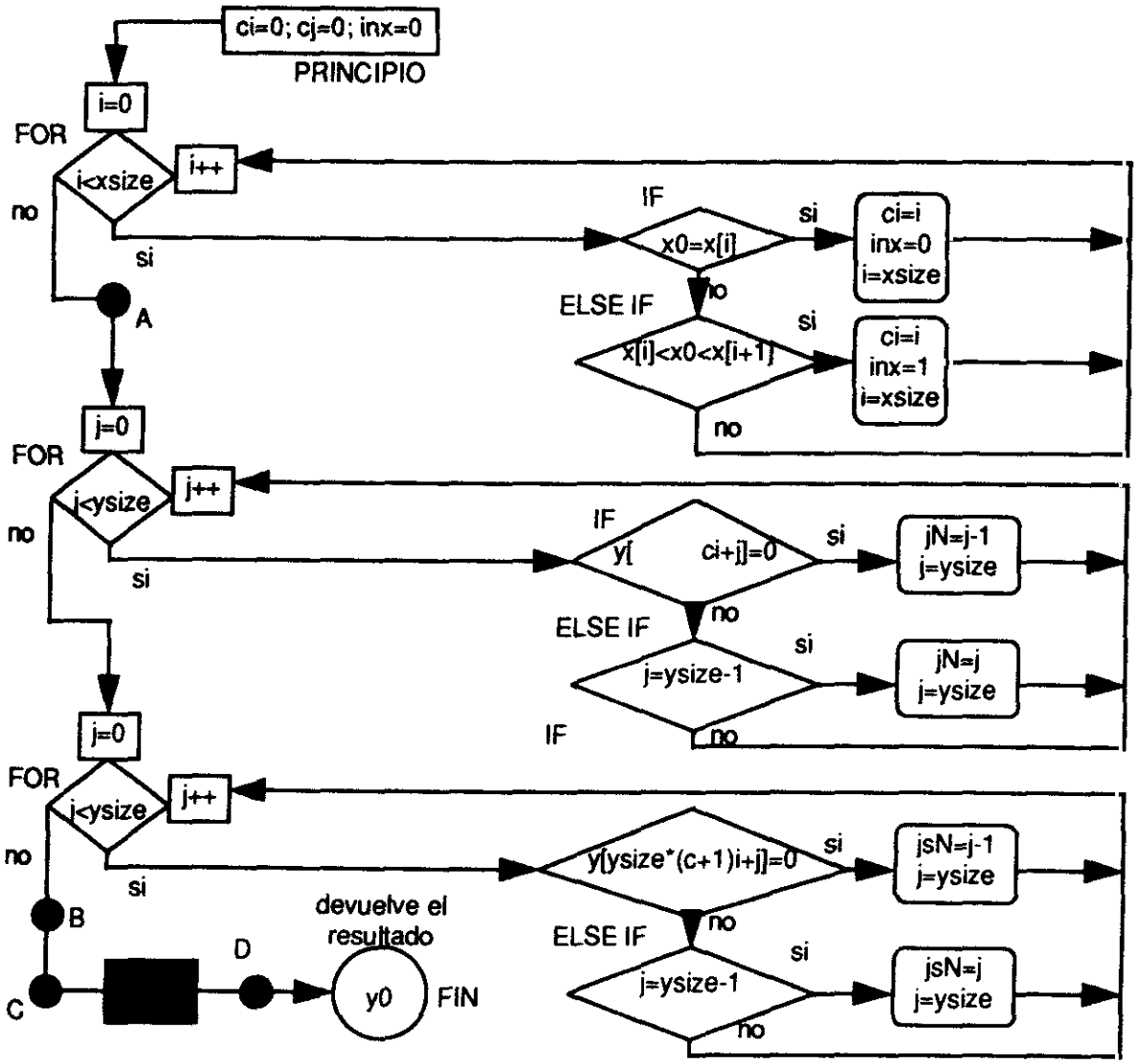

ci=0; cj=0; inx=0
PRINCIPIO
i=0
FOR
i<xsize
i++
no
si
IF
x0=x[i]
si
ci=i
inx=0
i=xsize
no
ELSE IF
x[i]<x0<x[i+1]
si
ci=i
inx=1
i=xsize
no
A
j=0
FOR
j<ysize
j++
no
si
IF
y[ci+j]=0
si
jN=j-1
j=ysize
no
ELSE IF
j=ysize-1
si
jN=j
j=ysize
IF
no
j=0
FOR
j<ysize
j++
no
si
y[ysize*(c+1)i+j]=0
si
jsN=j-1
j=ysize
no
ELSE IF
j=ysize-1
si
jsN=j
j=ysize
no
B
C
D
devuelve el
resultado
y0
FIN

Figura 5.33

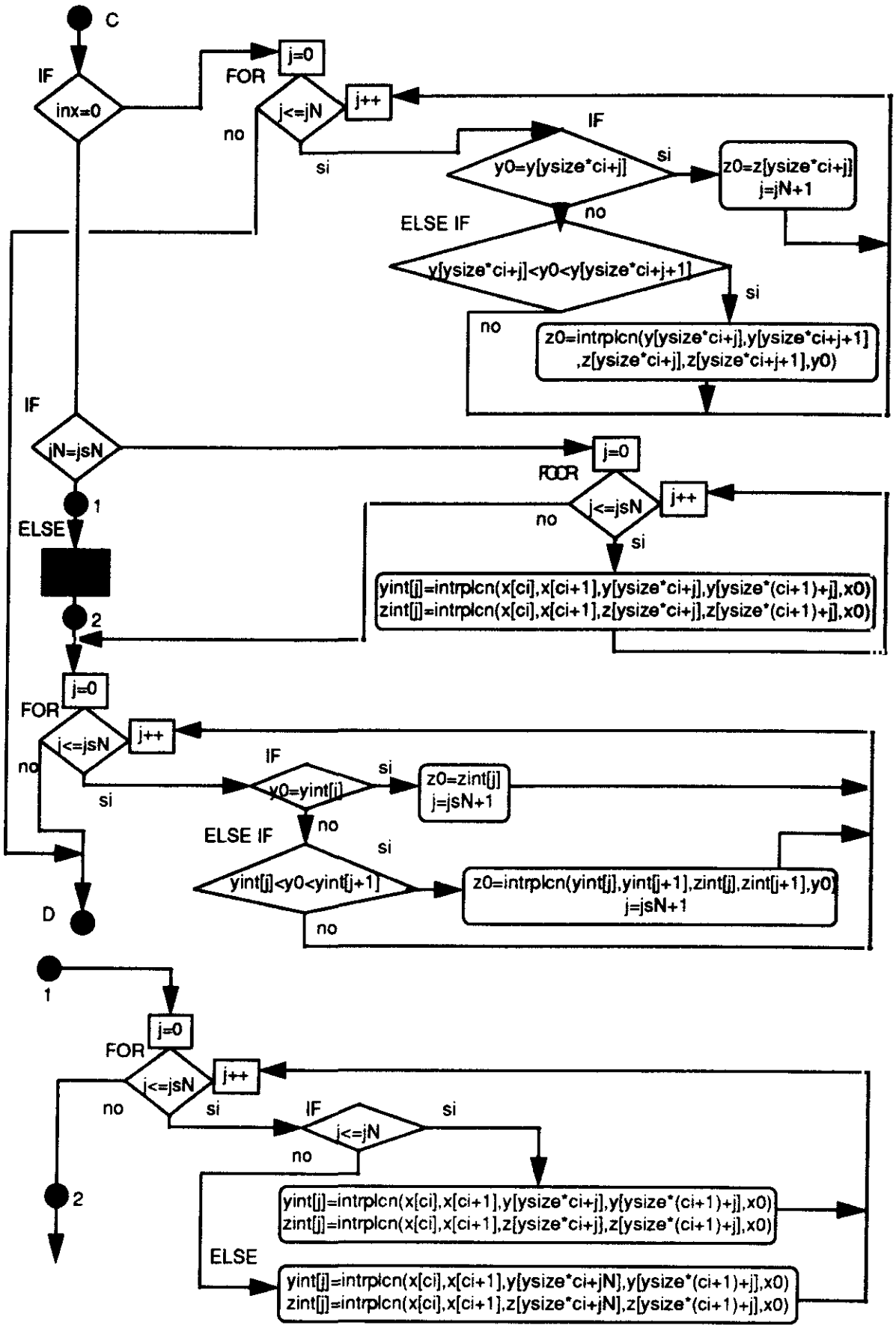

C
IF
inx=0
FOR
j=0
j<=jN
j++
no
si
IF
y0=y[ysize*ci+j]
si
z0=z[ysize*ci+j]
j=jN+1
no
ELSE IF
y[ysize*ci+j]<y0<y[ysize*ci+j+1]
si
no
z0=intrplcn(y[ysize*ci+j],y[ysize*ci+j+1]
,z[ysize*ci+j],z[ysize*ci+j+1],y0)
IF
jN=jsN
j=0
FOR
j<=jsN
j++
no
si
1
ELSE
yint[j]=intrplcn(x[ci],x[ci+1],y[ysize*ci+j],y[ysize*(ci+1)+j],x0)
zint[j]=intrplcn(x[ci],x[ci+1],z[ysize*ci+j],z[ysize*(ci+1)+j],x0)
2
j=0
FOR
j<=jsN
j++
no
si
IF
y0=yint[j]
si
z0=zint[j]
j=jsN+1
no
ELSE IF
yint[j]<y0<yint[j+1]
si
z0=intrplcn(yint[j],yint[j+1],zint[j],zint[j+1],y0)
j=jsN+1
no
D
1
j=0
FOR
j<=jsN
j++
no
si
IF
j<=jN
si
no
2
yint[j]=intrplcn(x[ci],x[ci+1],y[ysize*ci+j],y[ysize*(ci+1)+j],x0)
zint[j]=intrplcn(x[ci],x[ci+1],z[ysize*ci+j],z[ysize*(ci+1)+j],x0)
ELSE
yint[j]=intrplcn(x[ci],x[ci+1],y[ysize*ci+jN],y[ysize*(ci+1)+j],x0)
zint[j]=intrplcn(x[ci],x[ci+1],z[ysize*ci+jN],z[ysize*(ci+1)+j],x0)

Figura 5.34

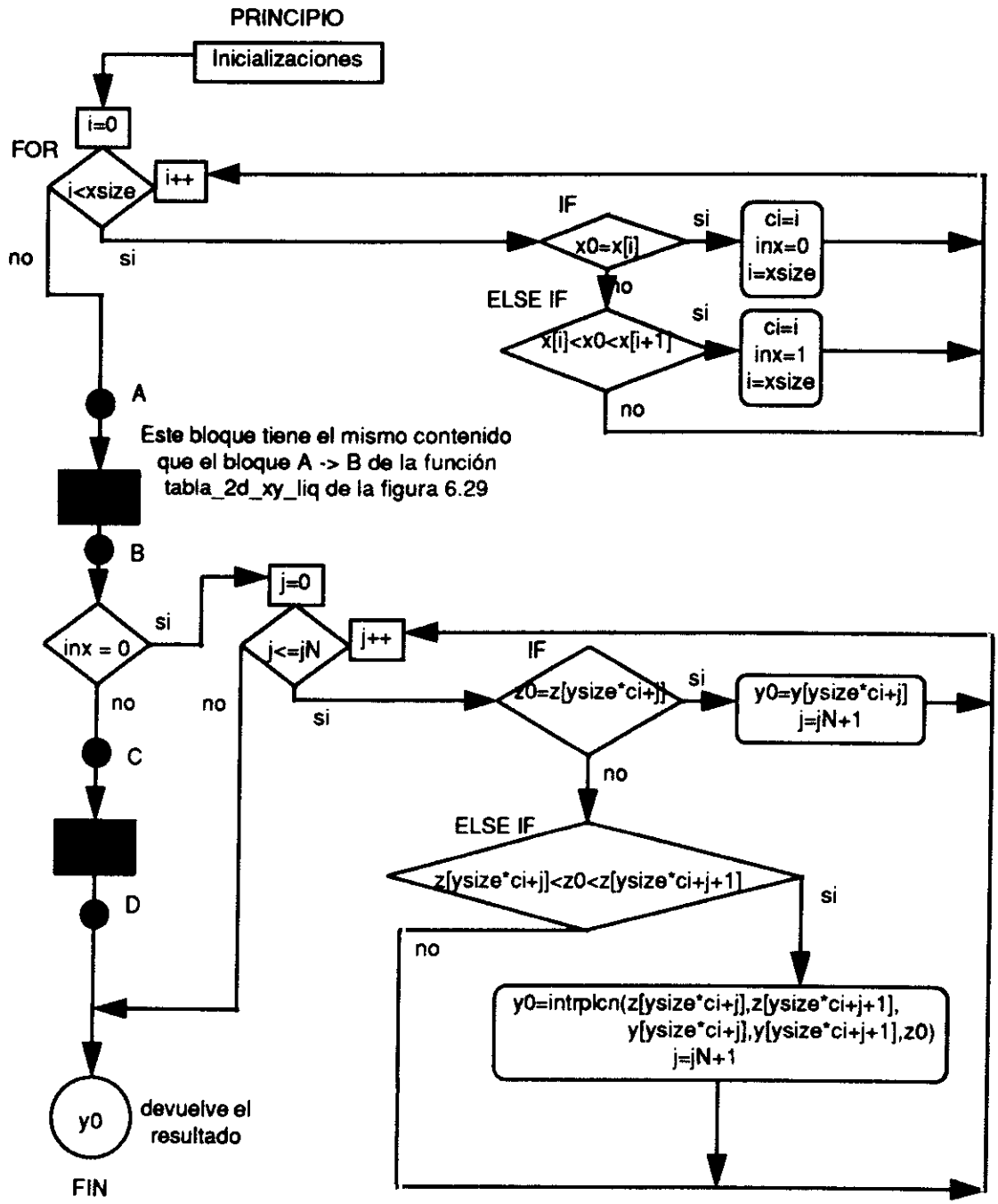

PRINCIPIO
Inicializaciones
i=0
FOR
i<xsize
i++
no
si
IF
x0=x[i]
si
ci=i
inx=0
i=xsize
no
ELSE IF
x[i]<x0<x[i+1]
si
ci=i
inx=1
i=xsize
no
A
Este bloque tiene el mismo contenido
que el bloque A -> B de la función
tabla_2d_xy_liq de la figura 6.29
B
inx = 0
si
no
j=0
j<=jN
j++
no
si
IF
z0=z[ysize*ci+j]
si
y0=y[ysize*ci+j]
j=jN+1
no
ELSE IF
z[ysize*ci+j]<z0<z[ysize*ci+j+1]
si
no
y0=intrplcn(z[ysize*ci+j],z[ysize*ci+j+1],
y[ysize*ci+j],y[ysize*ci+j+1],z0)
j=jN+1
C
D
y0
devuelve el
resultado
FIN

Figura 5.35

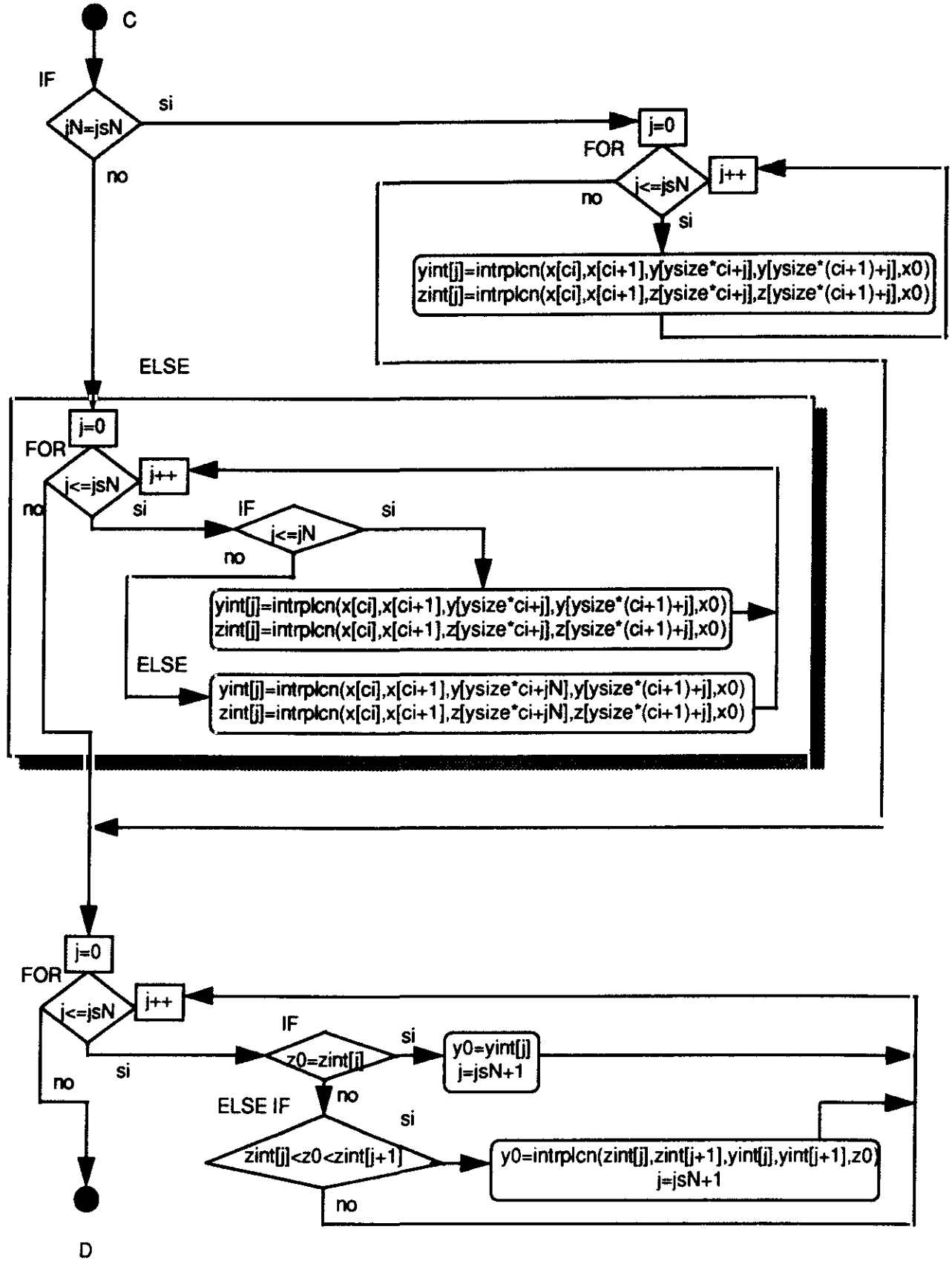

C
IF
jN=jsN
si
no
j=0
FOR
j<=jsN
j++
no
si
yint[j]=intrplcn(x[ci],x[ci+1],y[ysize*ci+j],y[ysize*(ci+1)+j],x0)
zint[j]=intrplcn(x[ci],x[ci+1],z[ysize*ci+j],z[ysize*(ci+1)+j],x0)
ELSE
j=0
FOR
j<=jsN
j++
no
si
IF
j<=jN
si
no
yint[j]=intrplcn(x[ci],x[ci+1],y[ysize*ci+j],y[ysize*(ci+1)+j],x0)
zint[j]=intrplcn(x[ci],x[ci+1],z[ysize*ci+j],z[ysize*(ci+1)+j],x0)
ELSE
yint[j]=intrplcn(x[ci],x[ci+1],y[ysize*ci+jN],y[ysize*(ci+1)+j],x0)
zint[j]=intrplcn(x[ci],x[ci+1],z[ysize*ci+jN],z[ysize*(ci+1)+j],x0)
j=0
FOR
j<=jsN
j++
si
no
IF
z0=zint[j]
si
y0=yint[j]
j=jsN+1
no
ELSE IF
zint[j]<z0<zint[j+1]
si
no
y0=intrplcn(zint[j],zint[j+1],yint[j],yint[j+1],z0)
j=jsN+1
D

Figura 5.36

V.3.3.- INTERPOLACIONES DIVERSAS.-

A continuación describiremos diversas interpolaciones para el tratamiento, tanto de las tablas de vapor, como de las distintas constantes requeridas en los modelos

a) Introducción:

A partir de una curva real que nos viene determinada por unas tablas de N datos (figura 5.37) queremos construir una función y(x) que nos interpole dichos valores. Esta función tendrá forma polinómica, con un orden n :

$$y(x) = a0+a1\,x+\ldots+an\,xn \quad \begin{cases} n = \text{orden de interpolación} \\ N = \text{Número de datos de la curva real a interpolar} \end{cases}$$

Hemos desarrollado una sencilla rutina encargada de este tipo de interpolaciones, cuyo cometido es el cálculo de los valores de ai. Estos coeficientes se hallan a partir de la matriz (A), de la siguiente forma:

$$\begin{vmatrix} s0 & s1 & \ldots & \ldots & sn \\ s1 & s2 & \ldots & \ldots & sn+1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ sn & sn+1 & \ldots & \ldots & s2n \end{vmatrix} * \begin{vmatrix} a0 \\ a1 \\ \ldots \\ \ldots \\ an \end{vmatrix} = \begin{vmatrix} t0 \\ t1 \\ \ldots \\ \ldots \\ tn \end{vmatrix}$$

donde:
$$\begin{cases} sk = \sum_{i=0}^{N} abcisas_i^k = abcisas_0^k + \ldots + abcisas_N^k \\ tk = \sum_{i=0}^{N} ordenadas_i \,.\, abcisas_i^k = ordenadas_0 \,.\, abcisas_0^k + \ldots + ordenadas_N \,.\, abcisas_N^k \end{cases}$$

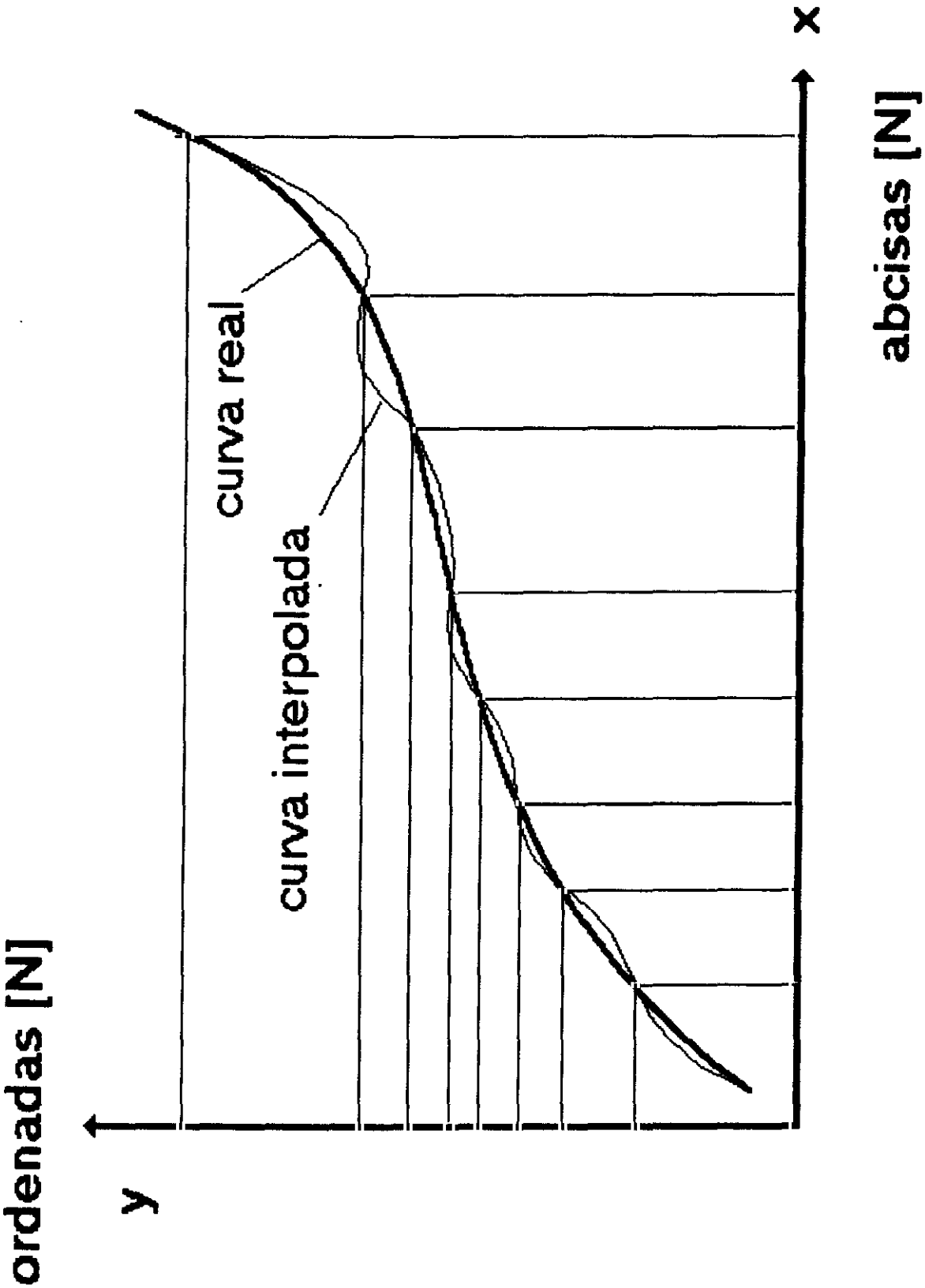


Figura 5.37

b) Interpolaciones del vapor:

Un estado de vapor viene siempre determinado por el concurso de, al menos, dos variables termodinámicas. La forma de hallar las restantes magnitudes puede hacerse, bien mediante ecuaciones y las tablas, o mediante curvas de vapor interpoladas. Ambos procedimientos serán empleados en nuestro programa.

Mediante ecuaciones y tablas.

Sabemos que la calidad de vapor, x, viene definida por :

$$x = \frac{h - h_l}{h_g - h_l} = \frac{v - v_l}{v_g - v_l} = \frac{s - s_l}{s_g - s_l}$$

donde h_l y h_g son las entalpías de los estados líquido saturado y vapor saturado, respectivamente, correspondientes a una presión igual a la del estado de vapor considerado, como puede verse en la figura 5.38. Lo mismo decimos para el volumen específico y la entropía. Así, por ejemplo, conociendo P y h, hallamos v y s de la siguiente manera:

$$\begin{cases} h_l = \text{tabla\_1d (P, h\_sat\_l, P\_sat, 76)} \\ v_l = \text{tabla\_1d (P, v\_sat\_l, P\_sat, 76)} \\ s_l = \text{tabla\_1d (P, s\_sat\_l, P\_sat, 76)} \end{cases} \quad \begin{cases} h_g = \text{tabla\_1d (P, h\_sat\_v, P\_sat, 76)} \\ v_g = \text{tabla\_1d (P, v\_sat\_v, P\_sat, 76)} \\ s_g = \text{tabla\_1d (P, s\_sat\_v, P\_sat, 76)} \end{cases}$$

$$\text{Dados} \begin{cases} P \\ h \end{cases} \rightarrow x = \frac{h - h_l}{h_g - h_l} \rightarrow \begin{cases} v = v_l + x\,(v_g - v_l) \\ s = s_l + x\,(s_g - s_l) \end{cases}$$

Recordemos que P_sat[76], T_sat[76], v_sat_l[76], h_sat_l[76], v_sat_v[76], v_sat_v[76] y h_sat_v[76] son los arrays en donde tenemos descritas las tablas de datos para los estados de líquido saturado (_sat_l) y vapor saturado (_sat_v).

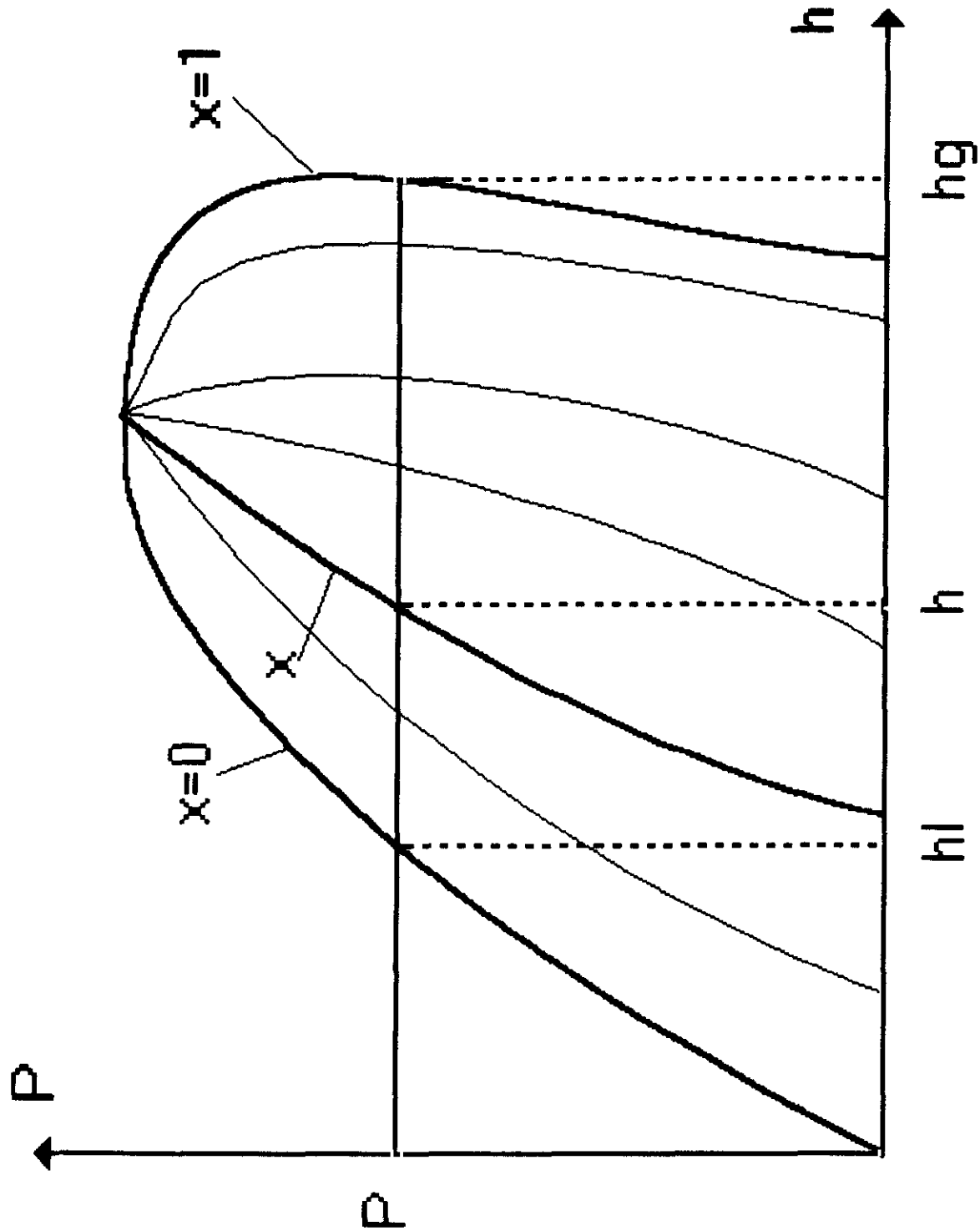


Figura 5.38

Mediante ecuaciones interpoladas.

Vamos a interpolar diversas curvas de entalpías y densidades correspondientes a una misma calidad de vapor x. Sea la siguiente tabla (array) de calidades :

j	0	1	2	3	4	5	6	7	8	9	10
x_dat[11]	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0

Con estos 11 valores de x vamos a construir las curvas interpoladas de h y d (densidad); si tuviésemos una x que no coincidiese con alguno de estos 11 valores, entonces construimos también las curvas de h y d correspondientes, mediante la interpolación de las curvas adyacentes, como enseguida se verá.

Para una x_dat[j] dada :

$$\begin{cases} h = c[0][j] + c[1][j]\ P + \ldots + c[orden][j]\ P^{orden} \\ \rho = a[0][j] + a[1][j]\ P + \ldots + a[orden][j]\ P^{orden} \end{cases}$$

Así pues, tenemos las siguientes dos tablas de coeficientes :

i \ j	0	1	2	...	...	...	10
0	c_{00}	c_{01}	...	...	...	...	...
1	c_{10}	c_{11}	...	...	...	...	...
2	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
orden	...	...	...	...	...	...	...

i \ j	0	1	2	...	...	...	10
0	a_{00}	a_{01}	...	...	...	...	...
1	a_{10}	a_{11}	...	...	...	...	...
2	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
orden	...	...	...	...	...	...	...

Para un valor de x cualquiera, que no sea ninguno de los x_dat[j], interpolaremos valores de las tablas c[i][j] y a[i][j].

Para calcular c[i][j], utilizamos la rutina descrita en la figura 5.39; básicamente, ésta consiste en, para una x_dat[j] dada, hallar hl y hg para cada valor de presión de la tabla de presiones correspondiente (P_sat[76]). Así, para cada P, con hl, hg y x_dat hallamos h. Una vez tengamos todos los puntos (todas las 76 entalpías), los interpolamos, mediante la función interpola anteriormente citada, que nos calculará los coeficientes cij.

En la figura 5.40 podemos ver el procedimiento que usaremos cuando tengamos una x cualquiera; calculamos los coeficientes c_x[i] interpolados de los correspondientes a los valores de x_dat anterior y posterior :

x dada -> $h = c\_x[0]+c\_x[1].P+...+c\_x[orden].P^{orden}$

A continuación ofrecemos las tablas de los valores obtenidos, para un **orden = 3.**

c_vap[4][11]

j \ i	0	1	2	3
0	334.578	0.275225	-2.23808e-5	6.19565e-10
1	565.849	0.253816	-2.07405e-5	5.68559e-10
2	797.12	0.232406	-1.91003e-5	5.17552e-10
3	1028.39	0.210996	-1.746e-5	4.66546e-10
4	1259.66	0.189586	-1.58198e-5	4.1554e-10
5	1490.93	0.168176	-1.41795e-5	3.64533e-10
6	1722.2	0.146767	-1.25392e-5	3.13527e-10
7	1953.48	0.125357	-1.0899e-5	2.62521e-10
8	2184.75	0.1039476	-9.25871e-6	2.11514e-10
9	2416.02	0.0825374	-7.61845e-6	1.60508e-10
10	2647.29	0.0611276	-5.97818e-6	1.09501e-10

a_vap[4][11]

j \ i	0	1	2	3
0	969.011	-0.0607167	4.67219e-6	-1.4474e-10
1	871.939	-0.0537018	4.1143e-6	-1.25608e-10
2	774.868	-0.0466869	3.55641e-6	-1.06476e-10
3	677.796	-0.0396721	2.99852e-6	-8.73436e-11
4	580.724	-0.0326572	2.44063e-6	-6.82115e-11
5	483.653	-0.0256424	1.88273e-6	-4.90794e-11
6	386.581	-0.0186275	1.32484e-6	-2.99473e-11
7	289.51	-0.0116126	7.66951e-7	-1.08152e-11
8	192.438	-0.00459776	2.0906e-7	8.31691e-12
9	95.3663	0.0024171	-3.48832e-7	2.74449e-11
10	-1.70535	0.00943197	-9.06723e-7	4.65811e-11

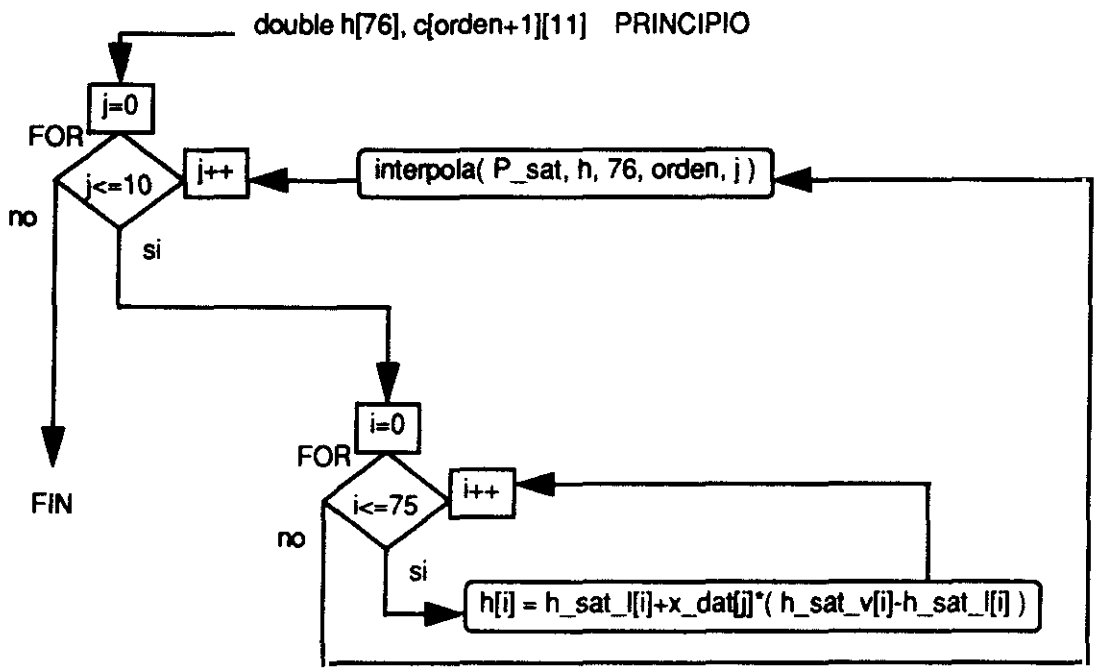

double h[76], c[orden+1][11] PRINCIPIO
j=0
FOR
j<=10
j++
interpola(P_sat, h, 76, orden, j)
no
si
FIN
i=0
FOR
i<=75
i++
no
si
h[i] = h_sat_l[i]+x_dat[j]*(h_sat_v[i]-h_sat_l[i])

Figura 5.39

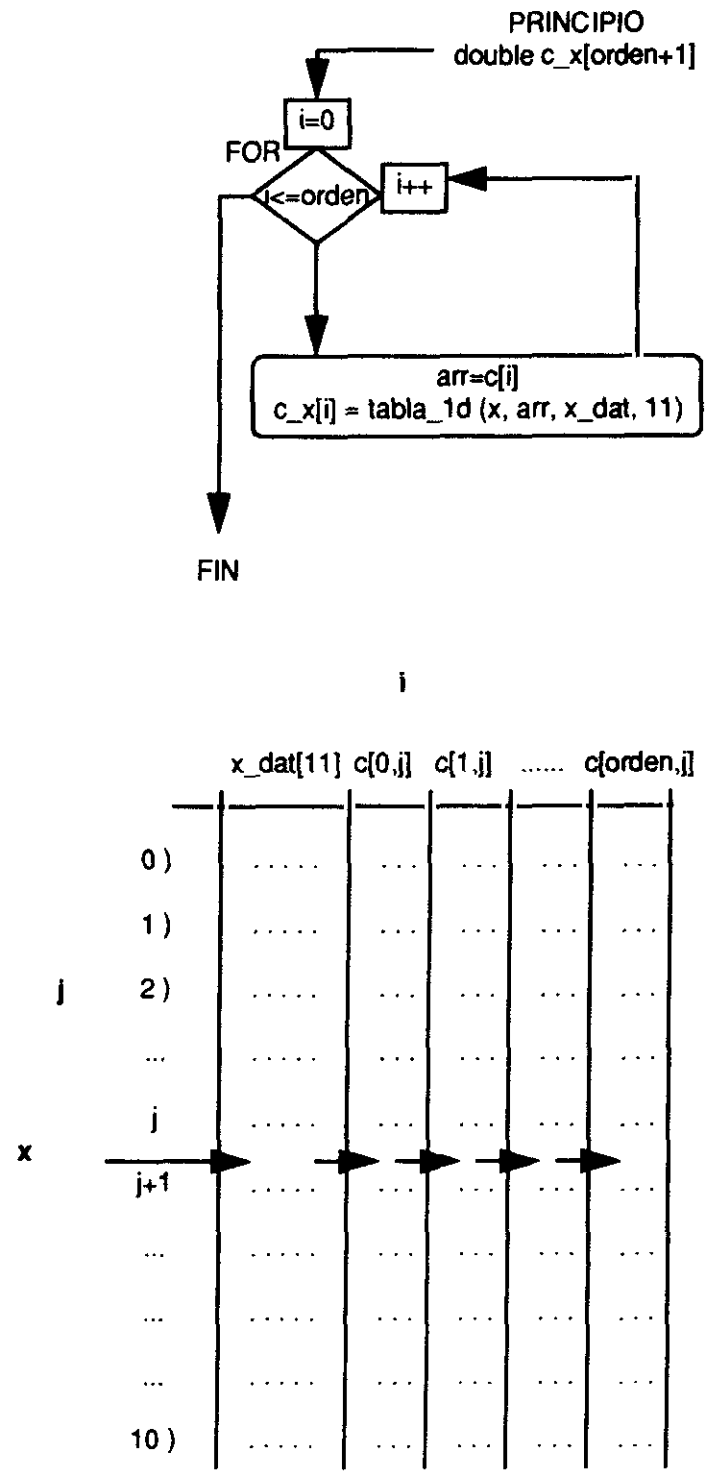

PRINCIPIO
double c_x[orden+1]
i=0
FOR
i<=orden
i++
arr=c[i]
c_x[i] = tabla_1d (x, arr, x_dat, 11)
FIN
i
x_dat[11] c[0,j] c[1,j] ..... c[orden,j]
0)
1)
j
2)
...
j
x
j+1
...
...
...
10)

Figura 5.40

c) Interpolación de las entalpías del estado líquido:

El estado líquido necesita, también, dos variables como mínimo para poder quedar completamente descrito. En nuestro modelo teórico, en ciertos momentos necesitaremos hallar las entalpías mediante interpolaciones. En esencia, para una T y una P dadas (ver la figura 5.41) queremos hallar la correspondiente h, de la siguiente manera :

$$h = e_0+e_1.P+...+e_n.P^n$$

Tenemos las siguientes tablas (arrays) :

```
P_l[27] = { ... , ... , ... , . . . . . , ... };
T_dat[22] = { 0, 10, 20, 30, 40, . . . . . , 1000 };
T_l[27][22] = {{ 0, 0.01 },                                   /* i=0 */
               { 0, 10, 17.5 },                               /* i=1 */
               { 0, 10, 20, 28.97 },                          /* i=2 */
               { 0, 10, 20, 30, 40, 40.3 },                   /* i=3 */
               . . . . . .
               { 0, 10, 20, 30, 40, . . . . . , 1000 } };     /* i=26 */
```

Por tanto, lo que haremos es calcular (para un orden = 3) los siguientes coeficientes:

		j						
		0	1	2	...	...	...	21
i	0	e_{00}	e_{01}	...	...	...	...	...
	1	e_{10}	e_{11}	...	...	...	...	...
	2	...	...	...	...	...	...	...
	3	...	...	...	...	...	...	...

En la figura 5.41 podemos observar el método que empleamos para calcular los coeficientes e[i][j]. La siguiente rutina se encarga de construir la anterior tabla de coeficientes:

```
for(i=0; i<=21; i++)            /* T_dat[i] */
        {
        k=0;
        for(jj=0; jj<=21; jj++)
                {
                for(ii=0;ii<=26;ii++)
                        {
                        if(T_l[ii][jj]==T_dat[i])
                                { k=ii; ii=26; jj=21; }
                        }
                }
        for(j=0;j<=26-k;j++)        /* P_l[ ] */
                {
                P_int[j]=P_l[k+j];
                h[j]=tabla_2d_xy_liq(P_l, 27, arrT_l, 22, arrh_l, P_l[k+j], T_dat[i]);
                }
        interpola(P_int, h, 26-k+1, orden, i); /* crea la tabla de coeficientes e[i][j] */
        }
```

Cuando tenemos una temperatura que no coincida con ninguna de las T_dat[i], entonces interpolamos de los coeficientes e[i][j], de una forma parecida a como comentábamos las interpolaciones de x para el caso de los estados de vapor.

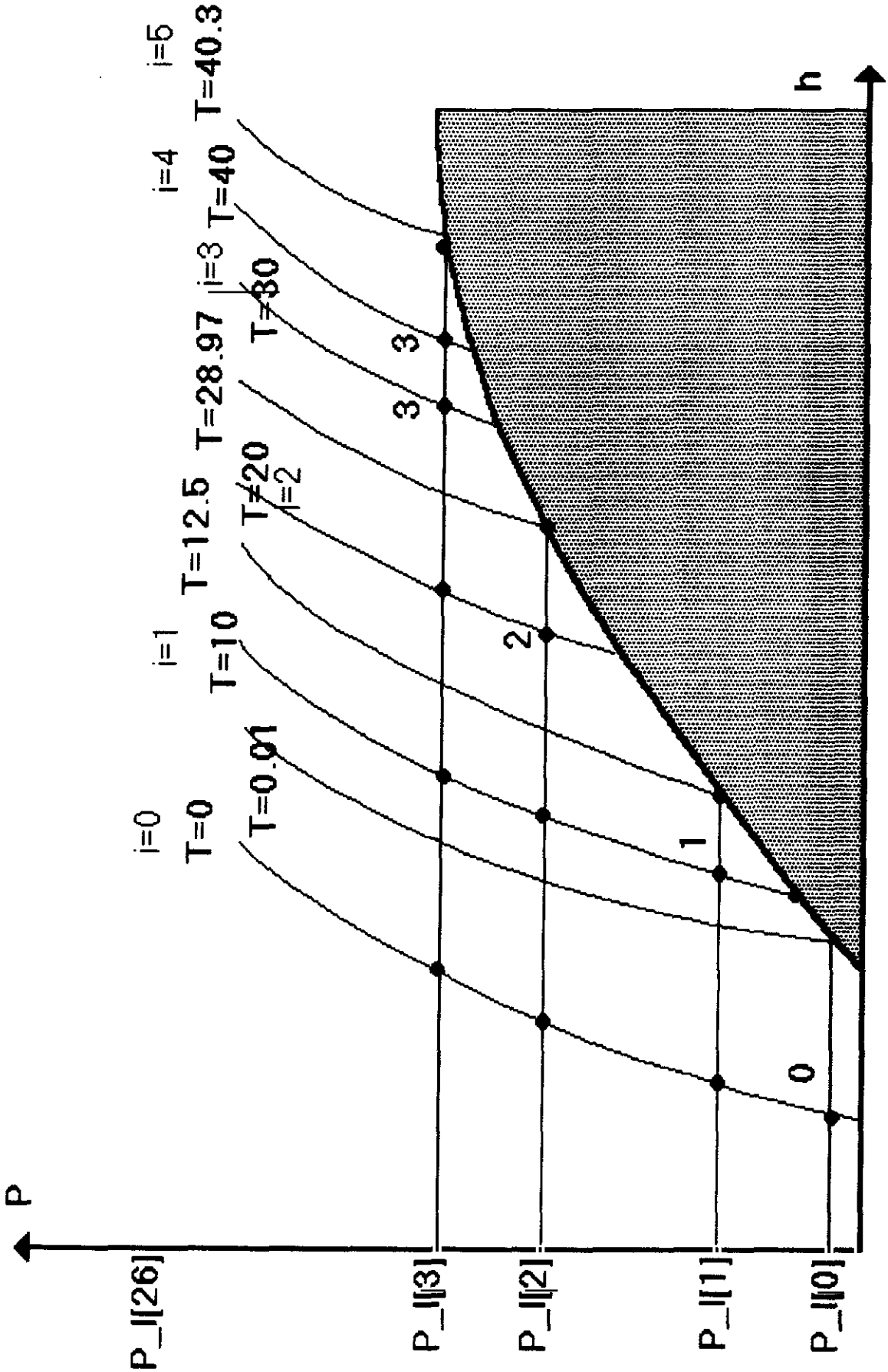

h
P
i=0
i=1
i=2
i=3
i=4
i=5
T=0
T=0.01
T=10
T=12.5
T=20
T=28.97
T=30
T=40
T=40.3
0
1
2
3
3
P_I[26]
P_I[3]
P_I[2]
P_I[1]
P_I[0]

Figura 5.41

Resultados (para un **orden = 3**) : **e_liq[4][22]**

	i			
	0	1	2	3
0	-0.0392145	0.00102177	-8.02481e-10	1.28784e-15
1	40.1942	0.00116187	-4.79947e-9	2.63826e-14
2	79.919	0.0013436	-9.51608e-9	5.53498e-14
3	119.179	0.00158071	-1.54134e-8	9.14079e-14
4	158.843	0.00177723	-2.03459e-8	1.21623e-13
5	199.676	0.00185452	-2.2542e-8	1.35125e-13
6	241.508	0.00183206	-2.24763e-8	1.34977e-13
7	282.052	0.00194509	-2.54558e-8	1.53251e-13
8	322.325	0.00209185	-2.9237e-8	1.76478e-13
9	364.262	0.00208397	-2.98682e-8	1.83819e-13
10	418.991	0.000750613	1.49423e-10	-5.42932e-16
j 11	524.901	0.000688171	3.76654e-10	-1.18711e-15
12	631.999	0.000618228	5.89228e-10	-1.61526e-15
13	740.75	0.000523101	1.1276e-9	-4.33679e-15
14	851.754	0.000403565	1.54434e-9	-3.9569e-15
15	1085.28	-8.60582e-5	5.92508e-9	-2.39363e-14
16	1353.82	-0.00138597	1.90047e-8	-7.73329e-14
17	1932.28	-0.0179914	2.79211e-7	-1.37379e-12
18	3358	-0.0614895	8.14286e-7	-3.5619e-12
19	3706	-0.00738223	-6.14286e-8	5.09524e-13
20	4159	-0.00457381	-9.28571e-9	9.52381e-14
21	4640.5	-0.00288024	-2.14286e-9	3.09524e-14

d) Interpolaciones de h_{vecd}, d_{vecd}, k_{31}, k_{11}, d_m, h_m, k_{12}, k_{32}, h_{rs} y k_5:

Los modelos teóricos nos piden el cálculo de distintas magnitudes y coeficientes mediante interpolaciones. Las siguientes serán dadas de forma más resumida, aprovechando la información ya obtenida con las anteriores interpolaciones.

$$\begin{cases} h_{vecd} = c_{01} + c_{11} P_{cd} + \ldots + c_{n1} P_{cd}^n \\ \rho_{vecd} = a_{01} + a_{11} P_{cd} + \ldots + a_{n1} P_{cd}^n \end{cases} \qquad \begin{cases} k_{31} = c_{11} + \ldots + n\, c_{n1} P_{cd}^{n-1} \\ k_{11} = a_{11} + \ldots + n\, a_{n1} P_{cd}^{n-1} \end{cases}$$

Así, dada x_{vecd}, hallamos los c_{ij} correspondientes, que con P_{cd}, nos lleva a obtener k_{31}. Análogamente, para k_{11}.

$$\begin{cases} d_m = a_{02} + a_{12} P_{cd} + \ldots + a_{32} P_{cd}^3 \\ h_m = c_{02} + c_{12} P_{cd} + \ldots + c_{32} P_{cd}^3 \end{cases} \qquad \begin{cases} k_{12} = a_{12} + \ldots + 3\, a_{32} P_{cd}^2 \\ k_{32} = c_{12} + \ldots + 3\, c_{32} P_{cd}^2 \end{cases}$$

h_m y d_m podemos hallarlas, también, mediante tablas. Finalmente, hallamos :

$$h_m = h_l + x_m(h_g - h_l) \qquad v_m = v_l + x_m(v_g - v_l) \qquad d_m = 1/v_m$$

Por último, en el caso del refrigerante, para los datos de entrada T_{re} y P_{cd}, hallamos d_{re} y h_{re} mediante tablas; para el refrigerante de salida, conociendo T_{rs} y P_{cd}, hallamos d_{rs} y h_{rs} mediante tablas, y k_5 así : $k_5 = e_1 + 2.e_2.P_{cd} + 3.e_3.P_{cd}^2$

Capítulo VI:

SIMULACION

Capítulo VI:

SIMULACION

Una vez que contamos con modelos, orientados a objeto, podemos realmente disfrutar combinando elementos, formando diversos ejemplos de plantas térmicas, y obteniendo información visual de cómo se comportan. En este Capítulo veremos cómo hemos realizado algunas simulaciones de gran impacto ilustrativo.

Ordenaremos la exposición mediante seis apartados.

En el primer apartado construiremos tres ejemplos de sistemas a simular, siguiendo una escala de complicación creciente.

Los dos apartados siguientes se dedican a presentar cómo se han coordinado las acciones, mediante un objeto que denominaremos Scheduler (acuñado en la terminología en inglés para simulaciones: podría quizá traducirse por Coordinador), y cómo se han efectuado aspectos concretos de la programación (gestión de ficheros, manejo de X-Windows, etc.).

A continuación, describimos las dos aplicaciones que hemos desarrollado, en sendos apartados. El primero de ellos se dedica a la versión sobre PC, con la que dió comienzo nuestro trabajo de programación. El segundo presenta la versión X-Window para estación de trabajo, que incluye un incremento sustancial de prestaciones.

Finalmente, exponemos en el sexto apartado algunos resultados de simulación, obtenidos mediante el uso de nuestras aplicaciones.

VI.1.- SIMULACIONES DE SISTEMAS TERMICOS.-

En el presente apartado expondremos tres sistemas térmicos relacionados con el ciclo de vapor, para su *simulación dinámica* (que reconoceremos por los nombres de sistema 1, sistema 2 y sistema 3). Son construcciones basadas en el nexado de dispositivos individuales cuyos modelos fueron descritos en el capítulo anterior.

Con el fin de facilitar la comprensión gradual de los fenómenos que se producen en un ciclo térmico, hemos decidido construir un primer sistema abierto mínimo (quemador, caldera); agregando un elemento supercalentador, construimos el segundo sistema, también abierto; por último el tercer sistema, ya cerrado, suma al segundo un condensador y una bomba. Deseamos un tipo de aprendizaje por exploración: mediante el manejo de estas sucesivas construcciones (incorporación de nuevos elementos), puede estudiarse mejor cada fenómeno, de forma que cabe identificar que un determinado comportamiento se debe a la actuación de un cierto elemento. Si mostrásemos sin más preámbulos el tercer sistema, sería más trabajoso identificar causas y efectos; por el contrario, sabiendo el comportamiento de un sub-sistema concreto, podremos estudiar más fácilmente el sistema completo.

Debido a la modularidad de estos sistemas (los elementos individuales están modelados por separado), y teniendo en cuenta las necesarias relaciones que deben darse entre algunos de ellos, es posible construir sistemas térmicos más complejos; la simulación que explicaremos sienta las bases para poder emprender estudios más complejos.

VI.1.1.- SISTEMA 1.-

En la figura 6.1 podemos observar el esquema de nuestro primer caso. Se trata de un sistema no cíclico, que dispone de cinco elementos: un quemador, una caldera, dos válvulas y una fuente de líquido.

Los elementos serán objetos en nuestro código C++. Así, tenemos los objetos siguientes:

fuente (de la clase *Fuente_de_liquido*),

valvula_2 (de la clase *Valvula*),

quemador_1 (de la clase *Quemador*),

cal (de la clase *Caldera*)

valvula_1 (de la clase *Valvula*).

La simulación parte de un instante t_0, y se calcula (o itera) cada h segundos, hasta un instante final t_f. Mediante las llamadas a los constructores de clase, no sólo creamos los objetos, sino que también inicializamos algunos de sus parámetros. De esta forma, con :

```
Caldera cal(0.2, 1., 0.5, P_caldera);
Quemador quemador_1(1.0);
Fuente_de liquido fuente(83.9);
Valvula valvula_1(0.005);
Valvula valvula_2(0.005);
```

estamos inicializando, en t_0, los parámetros R_c, L_c, n_c, P_c, Q_{ec}, h_{lec}, W_{vsc} y W_{lec} entendidos como valores privados o protegidos de cada objeto.

La manera de simular en el tiempo será la siguiente: cíclicamente, cada h segundos (empezando en t_0), el Scheduler hace la llamada

```
cal.set_variables( quemador_1.dar_Q(), valvula_2.dar_W(), fuente.dar_h(),
                   valvula_1.dar_W() );
```

para que el objeto *cal* pueda reconocer parámetros de otros objetos, necesarios para sus cálculos. Seguidamente, con la llamada

```
cal.calculos_t();
```

calcularemos los valores de diversos parámetros para el instante considerado (densidades, masas, constantes, temperaturas, etc., mediante las ecuaciones y tablas del modelo teórico); inmediatamente después, mediante la llamada

```
cal.inicializo_ncyPc();
```

calculamos los valores del nivel y la presión de la caldera en nuestro instante t. En este momento, conocemos los valores de todos los parámetros en t; para conocerlos en t+h, repetimos lo mismo; esquemáticamente, la simulación es así:

Datos iniciales: R_c, L_c, n_c, P_c, Q_{ec}, h_{lec}, W_{vsc} y W_{lec}

$$
\begin{cases}
t = t_0 \rightarrow \text{conozco} \begin{cases} \begin{cases} P_c \rightarrow \text{hallo} \begin{cases} k_1 , k_2 , k_3 , k_4 \\ h_{vc} , \rho_{vc} \\ h_{lc} , \rho_{lc} \end{cases} \\ n_c \rightarrow \text{hallo } V_{vc} , V_{lc} \end{cases} \rightarrow \text{hallo} \begin{cases} A_0 , A_1 , A_2 , A_3 , A_4 \rightarrow A , B \\ m_{vc}=\rho_{vc} V_{vc} \\ m_{lc}=\rho_{lc} V_{lc} \end{cases} \rightarrow m_c=m_{vc}+m_{lc} \\ W_{lec} , h_{lec} \\ W_{vsc} \end{cases} \\
t = t_0 + h \rightarrow \begin{cases} \begin{cases} \dfrac{dn_c}{dt} = A \rightarrow \displaystyle\int_{t_0}^{t} dn_c = \int_{t_0}^{t} A \, . \, dt \rightarrow \left[n_c = n_c + A \, . \, (t - t_0) \right] \\ \dfrac{dP_c}{dt} = B \rightarrow \displaystyle\int_{t_0}^{t} dP_c = \int_{t_0}^{t} B \, . \, dt \rightarrow \left[P_c = P_c + B \, . \, h \right] \end{cases} \rightarrow \\ \rightarrow \begin{cases} P_c \rightarrow \text{hallo} \ldots\ldots \\ n_c \rightarrow \text{hallo} \ldots\ldots \end{cases} \rightarrow \text{hallo } A_0 , A_1 , \ldots , A , B \quad , \quad m_{vc} , m_{lc} , m_c \\ W_{lec} , h_{lec} \\ W_{vsc} = W_{lec} - \dfrac{m_c|_{(t_0+h)} - m_c|_{(t_0)}}{h} \end{cases} \\
t = t + h = t_0 + h \rightarrow \text{etc.}
\end{cases}
$$

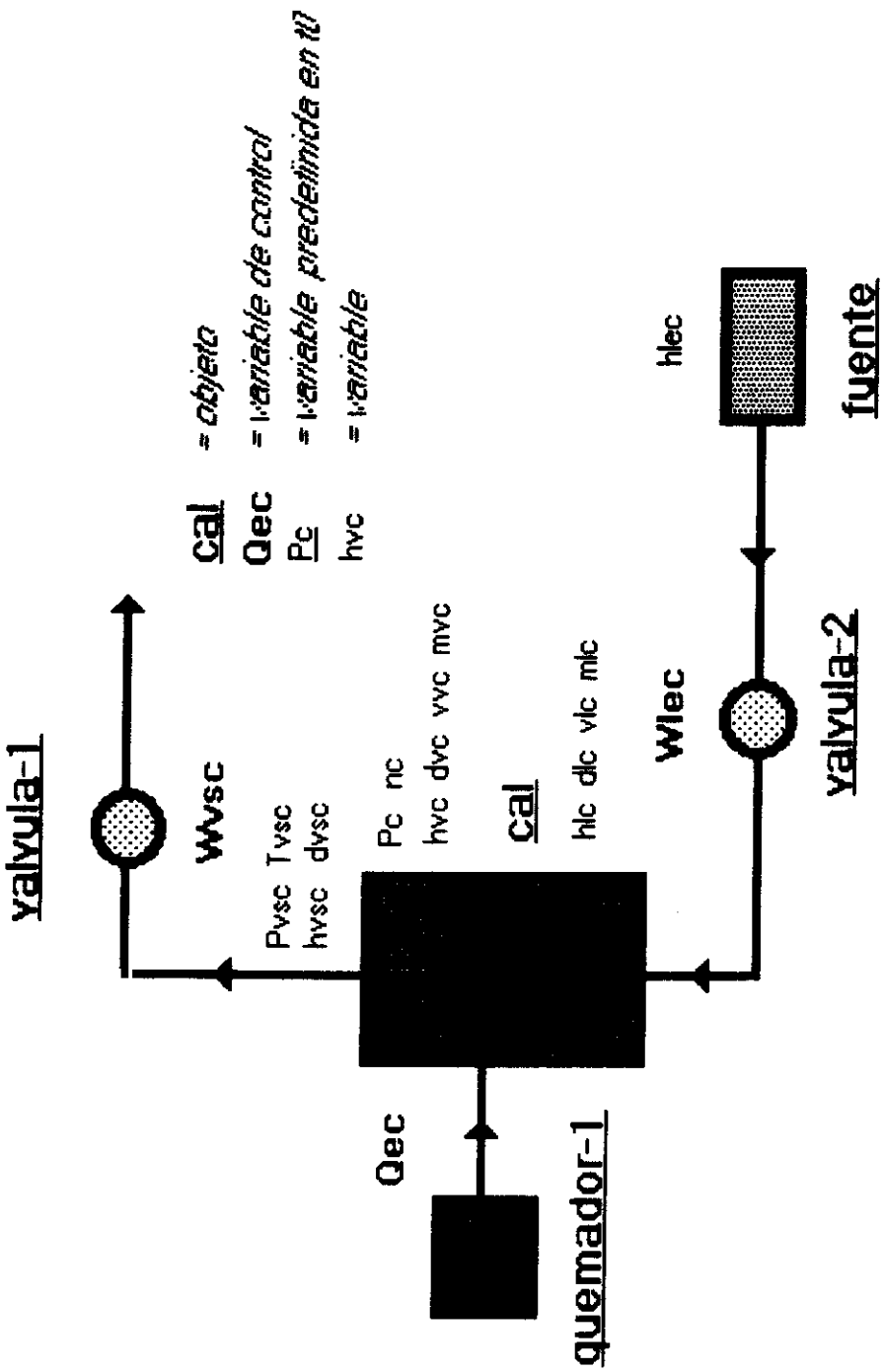

yalvula-1
Wvsc
Pvsc Tvsc
hvsc dvsc
Pc nc
hvc dvc vvc mvc
cal
hlc dlc vlc mlc
Wlec
hlec
fuente
yalvula-2
Qec
quemador-1
cal = objeto
Qec = variable de control
Pc = variable predetimida en t0
hvc = variable

Figura 6.1

VI.1.2.- SISTEMA 2.-

En la figura 6.2 podemos ver el esquema del sistema 2. Se trata de un sistema también abierto, que consta de ocho elementos: dos quemadores, una caldera, tres válvulas, un supercalentador y una fuente de líquido. Es, sencillamente, el sistema 1 al cual le hemos añadido un supercalentador al vapor saturado a la salida de la caldera.

Antes de describir la simulación del sistema, veamos cómo vamos a resolver las ecuaciones diferenciales para el supercalentador:

a) SUPERCALENTADOR:

- $t = t_0$:

  En el instante inicial hemos de conocer los parámetros Wves, hves, mvs, Qes y hvss. Con todos ellos, calculamos el valor de la constante del supercalentador:

$$k|_{t_0} = \left(h_{vss} - \frac{Q_{es} + h_{ves}\, W_{ves}}{W_{ves}} \right)_{t_0} . e^{\left(\frac{W_{ves}}{m_{vs}} t_0\right)} = k(t=t_0)$$

- $t = t_0 + h$:

  Considero que en el intervalo (t_0, t_0+h) los valores de Q_{es}, W_{ves} y h_{ves} permanecen constantes, a efectos del cálculo de h_{vss}. Si cambian de valor, consideraremos los nuevos valores de $t_0 + h$ cuando calculemos h_{vss} en t_0+2h. Por tanto, hallamos :

$$h_{vss}|_{t_0+h} = \left(\frac{Q_{es} + h_{ves}\, W_{ves}}{W_{ves}} \right)_{t_0} + k|_{t_0} . e^{-\left(\frac{W_{ves}}{m_{vs}}\right)_{t_0} . (t_0+h)}$$

  Ahora sí consideramos los nuevos valores de W_{ves}, h_{ves} y Q_{es} en t_0+h (nuevos valores pues estas magnitudes pueden variar en el tiempo). A continuación, calculamos:

$$k = \left(h_{vss} - \frac{Q_{es} + h_{ves}\, W_{ves}}{W_{ves}} \right)_{t_0+h} . e^{\frac{W_{ves}}{m_{vs}} (t_0+h)} = k(t=t_0+h)$$

- $t = t_0 + 2h$:

  Repetimos el mismo procedimiento:

$$h_{vss}|_{t_0+2h} = \left(\frac{Q_{es} + h_{ves}\, W_{ves}}{W_{ves}}\right)_{t_0+h} + k|_{t_0+h} \cdot e^{-\left(\frac{W_{ves}}{m_{vs}}\right)_{t_0+h} \cdot (t_0+2h)}$$

  Consideramos los nuevos (si es que varían) valores de W_{ves}, h_{ves} y Q_{es} en t_0+2h, y a continuación, calculamos nuevamente k:

$$k = \left(h_{vss} - \frac{Q_{es} + h_{ves}\, W_{ves}}{W_{ves}}\right)_{t_0+2h} \cdot e^{\frac{W_{ves}}{m_{vs}}(t_0+2h)}$$

  Y así, sucesivamente.

b) SISTEMA 2:

Cuando el Scheduler crea los objetos, se inicializan los siguientes valores (en t_0) : n_c, P_c, Q_{ec}, W_{lec}, hlec, W_{vsc}, Q_{es}, h_{vss}, W_{vss} y m_{vs}. De éstos, Q_{ec}, W_{lec}, W_{vsc} y Q_{es} son variables de control (que nosotros podremos cambiar a gusto durante la simulación).

```
Caldera cal(0.2, 1., 0.5, P_caldera);
Quemador quemador_1(1.0);
Quemador quemador_2(5.0);
Fuente_de liquido fuente(83.9);
Valvula valvula_1(0.005);
Valvula valvula_2(0.005);
Valvula valvula_3(valvula_2.dar_W());
Supercalentador sup(2850., 1., 10.);
```

La simulación la realizará el Scheduler de una forma cíclica, instante por instante, partiendo de t_0 hasta t_f cada h segundos. Supongamos, pues, que comenzamos a iterar (estamos en t_0). En primer lugar, y mediante:

```
cal.set_variables( quemador_1.dar_Q(), valvula_2.dar_W(), fuente.dar_h(),
                   valvula_1.dar_W() );
```

asignamos las entradas a la caldera Q_{ec}, W_{lec}, hl_{ec} y la salida W_{vsc}. Después, mediante:

```
cal.calculos_t()
```

hallamos h_{vc}, h_{lc}, d_{vc}, d_{lc}, V_{vc}, Vl_c, m_{vc}, m_{lc}, A y B, pues en este momento conocemos los valores n_c, P_c, W_{lec}, W_{vsc}, h_{lec} y Q_{ec}. A continuación, asigno las entradas al supercalentador, P_{ves}, T_{ves}, h_{ves}, v_{ves}, Q_{es}, W_{ves} y s_{ves} mediante la llamada

```
sup.set_variables(cal.dar_Pc(),cal.dar_tc(),cal.dar_hvc(),1./cal.dar_dvc(),
                  quemador_2.dar_Q(),valvula_1.dar_W(),cal.dar_svc() );
```

para después, y mediante el método

```
sup.calculos_t( t,sup.dar_Wves() );
```

poder calcular T_{vss}, P_{vss} v_{vss} y k en el instante considerado, t_0. Por tanto, tenemos ahora todas las variables determinadas en t_0. Vamos a continuación, y antes de repetir el ciclo, a calcular n_c, P_c y h_{vss} para el instante posterior, t_0+h; esto es así pues considero que en el intervalo de tiempo entre t_0 y t_0+h, las variables necesarias para el cálculo de n_c, P_c y h_{vss} son constantes con el valor que tenían en t_0. Estos cálculos se realizan mediante :

```
cal.inicializo_ncyPc(h);
sup.inicializo_hvss(t+h);
```

en las cuales se están resolviendo las ecuaciones diferenciales de la siguiente forma:

$$\begin{cases} n_{c|_{t_0+h}} = n_{c|_{t_0}} + A|_{t_0}\, h \\ P_{c|_{t_0+h}} = P_{c|_{t_0}} + B|_{t_0}\, h \\ h_{vss|_{t_0+h}} = \dfrac{Q_{es} + h_{ves}\, W_{ves}}{W_{ves}}\Big|_{t_0} + k|_{t_0} \cdot e^{-\frac{W_{ves}}{m_{vs}}|_{t_0} \cdot (t_0+h)} \end{cases}$$

Ahora modificamos las variables de control a unos nuevos valores que serán tomados como valores para t_0+h. Después de ésto, comenzamos de nuevo por la primera llamada, para el instante t_0+h, y así sucesivamente.

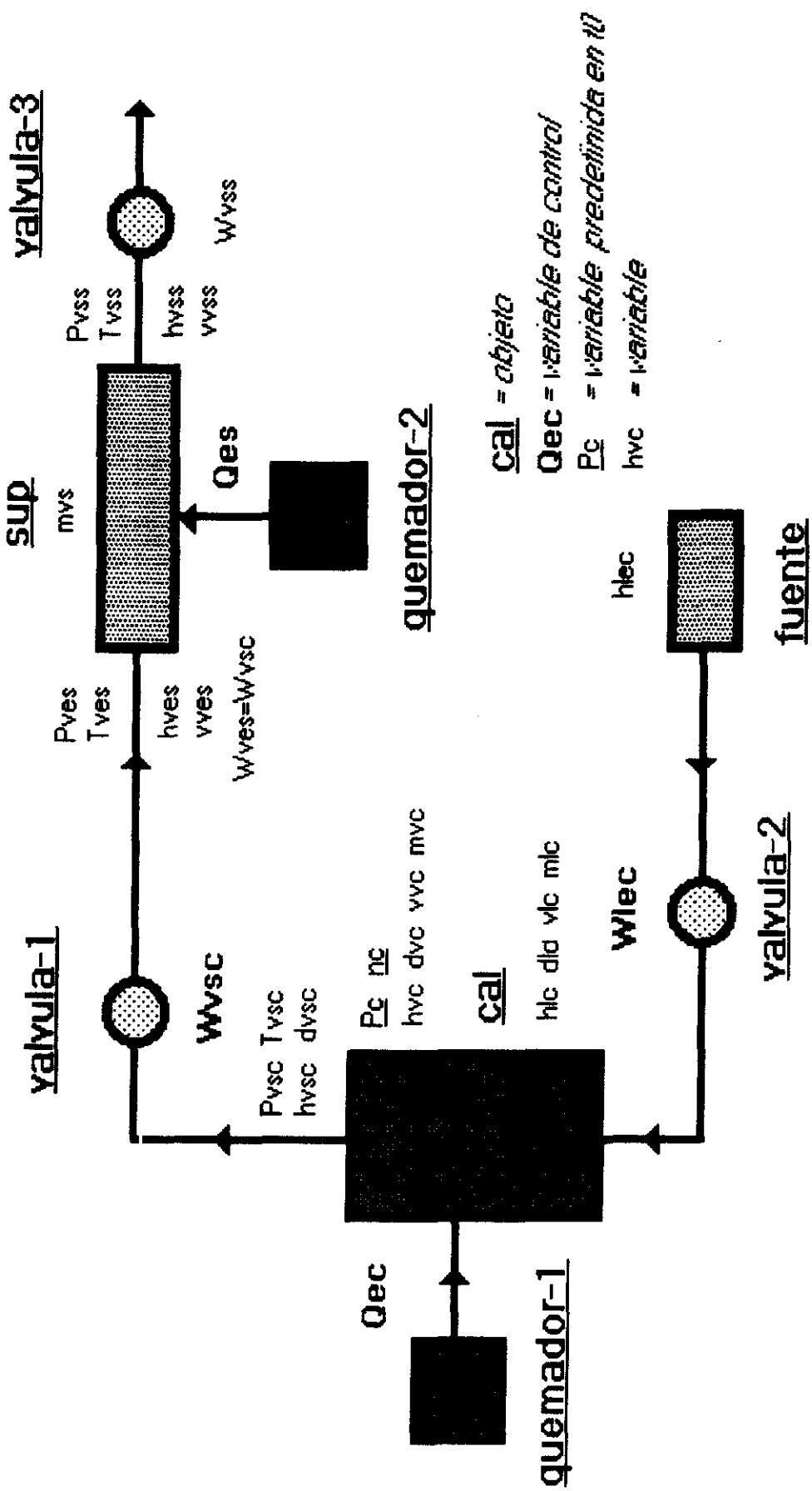

valvula-3
Pvss
Tvss
hvss
vvss
Wvss
sup
mvs
Qes
quemador-2
cal = objeto
Qec = variable de control
Pc = variable predefinida en tl
hvc = variable
Pves
Tves
hves
vves
Wves=Wvsc
hlec
fuente
valvula-1
Wvsc
Pvsc Tvsc
hvsc dvsc
Pc nc
hvc dvc vvc mvc
cal
hlc dlc vlc mlc
Wlec
valvula-2
Qec
quemador-1

Figura 6.2

VI.1.3.- SISTEMA 3.-

Como podemos apreciar en la figura 6.3, se trata de un esquema cíclico, que representa nuestro deseado prototipo de planta de vapor. Tiene todos los elementos necesarios para el típico ciclo de vapor, y consiste en una caldera, un supercalentador, una turbina, un condensador, una bomba, una válvula y dos quemadores.

También podemos apreciar que se trata del sistema 2, el cual ha sido cerrado mediante la adición de una turbina, condensador y bomba. Los resultados de la simulación de los anteriores sistemas servirán tan sólo para no perderse en la simulación del presente sistema, y así tener una visión más didáctica de las causas y efectos de los eventos que se producen.

En este sistema, podemos distinguir tres clases de variables:

- *Variables de control :*
  Q_{ec}, W_{vsc}, Q_{es}, k_x, W_b, W_r, T_{re} y T_{rs}. Estas variables cambiarán sus valores a lo largo del tiempo a voluntad nuestra; mediante ellas podremos controlar numerosos procesos, evitando comportamientos no deseados.

- *Variables iniciales fijas :*
  R_c, L_c, m_{vs}, k_{sup}, m_r, R_{cd} y L_{cd}. Estas variables toman sus valores inicialmente (a través de la creación de los objetos, por ejemplo), de manera que no cambian sus valores en todo el proceso de simulación.

- *Variables de estado que se inicializan y cambian :*
  P_c, n_c, h_{vss}, h_{cd} y P_{cd}. Estas variables (de gran importancia) varían con el tiempo. Sus valores iniciales han de ser conocidos con el objeto de poder resolver las distintas ecuaciones diferenciales.

A continuación explicaremos la forma de simular en el tiempo este sistema, resolviendo las numerosas ecuaciones del modelo matemático.

En primer lugar, y antes de entrar a iterar, creamos una serie de objetos que, a su vez, inicializarán algunos valores que serán usados en $t=t_0$:

Caldera **cal** (0.2, 1., 0.5, P_caldera);	Establezco : R_c, L_c , P_c y n_c.
Quemador **quemador_1** (1.0);	Establezco : Q_{ec}.
Valvula **valvula_1** (0.005);	Establezco : W_{vsc}.
Supercalentador **sup** (2850., 1., 10.);	Establezco : h_{vss}, m_{vs}, k_{sup}.
Quemador **quemador_2** (5.0);	Establezco : Q_{es}.
Turbina **turb** (0.1);	Establezco : K_x.
Condensador **cond** (0.2, 1., 0.5, 2.34, 0.005, 1., 20., 25);	Establezco : R_{cd}, L_{cd}, n_{cd}, P_{cd}, W_r, m_r, T_{re} y T_{rs}.
Bomba **bomb** (0.005);	Establezco : W_b.

- t = t0

  — cond.set1_variables();
  Con esta llamada, se conocen los valores en este instante t de k_2, k_4, V_{lcd}, V_{vcd}, d_{lcd}, h_{lcd}, h_g, h_l, v_g y v_l.

  — bomb.set_variables(cond.dar_Pcd(),cond.dar_dlcd(),cal.dar_Pc());
  Ahora establecemos P_{leb}, d_{leb}, P_{lsb}, d_{lsb}, T_{lsb} y h_{lsb}.

  — cal.set_variables(quemador_1.dar_Q(),bomb.dar_Wb(),
  bomb.dar_hlsb(),valvula_1.dar_W());
  Establezco así Q_{ec}, W_{lec}, h_{lec} y W_{vsc}.

Pasada esta primera etapa de inicialización de valores en t, comenzamos con los cálculos mediante ecuaciones y tablas :

  — cal.calculos_t();
  Calculamos los valores de k_1, k_2, k_3, k_4, d_{vc}, h_{vc}, v_{vc}, s_{vc}, m_{vc}, d_{lc}, h_{lc}, v_{lc}, s_{lc}, m_{lc}, m_c, t_c, A_0, A_1, A_2, A_3, A_4, A y B.

  — sup.set_variables(cal.dar_Pc(),cal.dar_tc(),cal.dar_hvc(),
  1./cal.dar_dvc(),quemador_2.dar_Q(),valvula_1.dar_W(),cal.dar_svc());
  Se establecen las variables P_{ves}, T_{ves}, h_{ves}, v_{ves}, Q_{es}, W_{ves} y s_{ves}.

  — sup.calculos_t(t,sup.dar_Wves());
  Calculamos P_{vss}, W_{vss}, T_{vss}, v_{vss}, s_{vss} y k.

— turb.set_variables(sup.dar_Wvss(),sup.dar_Pvss(),sup.dar_Tvss(), sup.dar_hvss(),sup.dar_svss(),sup.dar_vvss(),cond.dar_Pcd());
Ahora establecemos las variables W_{vet}, P_{vet}, T_{vet}, h_{vet}, s_{vet}, v_{vet}, P_{vst}, R_i, W_{tx}, W_{vst}, h_{vsti}, h_{vst}, T_{vst}, x_{vst}, v_{vst} y s_{vst}.

— cond.set2_variables(turb.dar_Wvst(),turb.dar_Tvst(),turb.dar_xvst(), turb.dar_hvst(),turb.dar_vvst(),bomb.dar_Wb());
Ahora establezco W_{lscd}, W_{vecd}, T_{vecd}, x_{vecd}, h_{vecd}, d_{vecd},T_{cd}, m_{vecd}, c_x[i], a_x[i], k_{31}, k_{11}, x_m, T_m, h_m, d_m, m_m, c_x[i], a_x[i], k_{12}, k_{32}, h_{re}, d_{re}, d_{rs}, h_{rs}, e_T[i], k_5, A_0, A_1, A_2, A_3, A_4, A_{4p}, A y B del condensador.

A continuación, hacemos una serie de cálculos que nos inicializan para t_0+h las variables n_c, P_c, h_{vss}, n_{cd} y P_{cd}, mediante :

— cal.inicializo_ncyPc(h);

— sup.inicializo_hvss(t+h);

— cond.inicializo_ncdyPcd(h);

Por último, variamos (si así lo deseamos) los parámetros de control, cuyos nuevos valores son los que consideramos que tienen ya en t+h.

- t = t0+h :

Repetimos las mismas llamadas, y así sucesivamente para todo t_0 + 2h, etc.

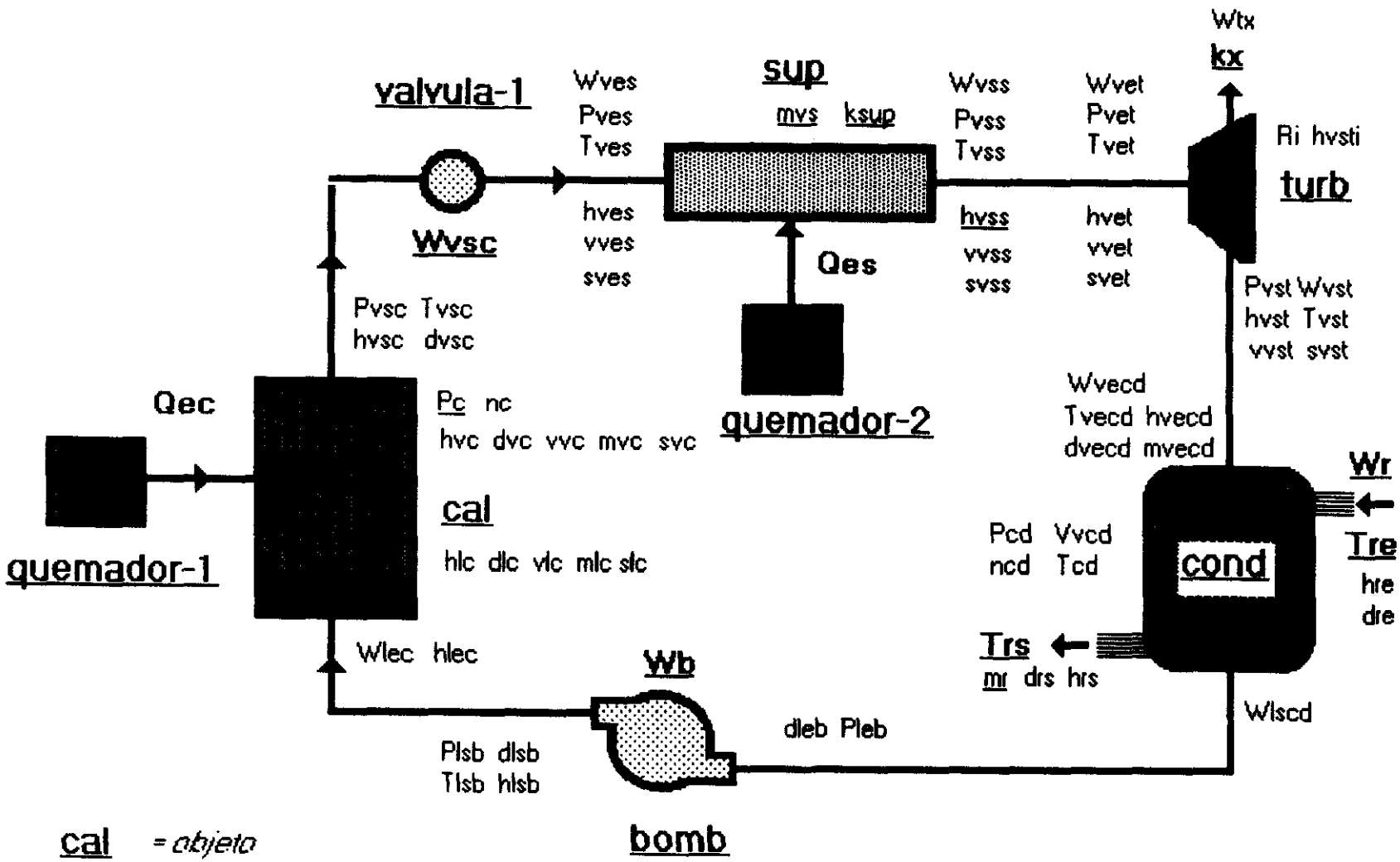


Figura 6.3

cal = *objeto*
Qec = *variable de control*
Pc = *variable predefinida en tl*
hvc = *variable*

VI.2.- EL SCHEDULER (COORDINADOR).-

En este apartado trataremos la concepción del Scheduler, y sus diversas actuaciones en la simulación de nuestros sistemas. Conviene precisar ahora que fué usado para desarrollar las dos versiones programadas (para PC y para X-Windows), manteniendo ligeras diferencias en cuanto a sus facultades.

El concepto de Scheduler (o coordinador, llamémoslo también así en una de sus posibles traducciones), surge al plantearse la forma de organizar la simulación en nuestro programa. Así, se ha de pensar en numerosas cuestiones, como el decidir qué tipo de alcance han de tener ciertas variables (localidad, globalidad), qué jerarquía de valores han de tener las funciones, qué rutinas han de controlar qué cosas, etc.

Tareas tan diversas como llevar el cómputo del tiempo, pintar unos ejes, comprobar sobre qué ventana está el ratón, abrir un fichero, etc., nos pueden llevar a una cierta desorganización, sobre todo cuando el número de tareas a desarrollar es bastante elevado. Por otra parte, queremos dejar el código del programa lo suficientemente claro y comprensible para facilitar su estudio por personas que no hayan participado en la elaboración de dicho código.

Por otro lado, cuando nos introducimos en la Programación Orientada a Objeto, surgen tareas y conceptos que han de manejarse con especial cuidado, pues habrá que determinar qué tipo de accesibilidad tienen partes de estos objetos, y por parte de quién.

Para facilitar la manejabilidad del código y repartir las tareas de una forma más natural (acorde con el razonamiento humano), se ha planteado la figura del Scheduler. Este "ente" no es más que un objeto de una clase (programación orientada a objeto), que se encargará de coordinar las tareas a desarrollar por objetos de otras clases, funciones, bucles, etc., en la totalidad del programa.

En la figura 6.4 podemos observar la filosofía de este coordinador. Las distintas tareas a realizar no tienen porqué relacionarse entre sí (llamarse unas a otras, pedirse datos, etc.), sino que todo se hace a través del Scheduler. Así pues, si por ejemplo una tarea necesita conocer el instante en que nos encontramos, se lo dará el Scheduler, y

no la rutina encargada de llevar el tiempo. Por tanto, el Scheduler coordina diciendo "ahora haces tú ésto, y a continuación, tú lo otro".

Evidentemente, aparte de coordinar, el Scheduler necesita tener cierta autoridad como para ordenar las líneas generales de actuación. Las actuaciones de menor importancia las llevarán a cabo cada tarea correspondiente.

Como hemos dicho, el Scheduler es una clase. Un objeto de dicha clase, por ejemplo "Juan", será el encargado de coordinar la simulación. Tendrá diversas variables y funciones con el objeto de hacer todo ésto. Una peculiaridad es que todas sus variables y funciones serán de caracter protegido, es decir, no accesible por ningún otro objeto o función del programa. Tan sólo una función, start(), será pública; llamando desde main() :

```
Juan.start();
```

se accede a este método, el cual va a comenzar el proceso de simulación de una forma coordinada, llamando a sus propias funciones que tiene protegidas.

Las principales tareas y características que coordina el scheduler, y que pueden apreciarse resumidas en la figura 6.5, son :

- **Tiempo.**
  El Scheduler controla el tiempo de la simulación, t. Cuenta desde un instante inicial t0 hasta uno final, tf, cada h segundos.

- **Gestión de menús y ventanas.**
  Controla el flujo de los distintos menús, es decir, qué ventanas de opciones se despliegan al hacer qué cosas; reconoce qúe eventos producen la apertura y cierre de ventanas; etc. Las variables 01,02, ..., o6 identifican opciones de menús. Las rutinas abro_menu_...(...) , menu_...() y cierro_menu_...() se encargan de abrir, gestionar y cerrar las distintas ventanas que configuran los menús iniciales, de datos, de ficheros, de controles, etc.

  Por otra parte, se encarga de reconocer los distintos eventos (de teclado y de ratón) con el fin de interpretarlos y actuar en consecuencia (así por ejemplo, si el Scheduler vé cómo el ratón ha sido presionado cuando estaba sobre un botón de "cerrar", entonces cierra la ventana correspondiente"). Esto lo efectuará mediante funciones como raton_menu_...(...) ,raton_...(...), teclado_...(...), etc.

- **Simulación de sistemas.**
  Mediante tres métodos, esquema_1(), esquema_2() y esquema_3() llamará a simular los tres sistemas considerados. Cada una de estas funciones creará una serie de objetos y pondrá en marcha diversas variables y rutinas propias que serán las encargadas de simular, efectuar cálculos, grabar sobre ficheros, abrir gráficos, dibujar curvas, etc. Tan sólo una pequeña función, cambio_coordenadas(...) será propia del Scheduler.

- **Gestión de ficheros.**
  El scheduler abrirá ficheros (*fp, fh, *filename, etc.) para guardar o leer los datos de una simulación; escribirá sobre ellos, los cerrará, etc. Establece una numeración de ficheros (num_fich), qué número de datos y de qué forma agrupados (N_grupos_dat, hit, num_iter), cuándo y cómo se graba en ellos o se leen, etc.

- **Variables termodinámicas fundamentales.**
  Al Scheduler le interesa acceder en todo momento a una serie de variables termodinámicas de importancia en la simulación de los sistemas, debido a que ello le permite con más facilidad inicializar ciertos parámetros de la simulación.

Por último, presentamos a continuación el listado de la clase Scheduler, según aparece en la versión para X-Window del programa:

```
Class Scheduler
    {
    private:
    protected:
            char    theDisplayName[120];
            int     o1,o2,o3,o4,o5,o6;

            void abro_menu_inicial(void);
            int menu_inicial(void);
            int raton_menu_inicial(XButtonEvent *theEvent);

            void abro_menu_modo(void);
            int menu_modo(void);
            int raton_menu_modo(XButtonEvent *theEvent);
            void cierro_menu_modo(void);

            void abro_menu_parametros(void);
            int menu_parametros(void);
            int raton_menu_parametros(XButtonEvent *theEvent);
            void imprimo_parametros(void);
            void cierro_menu_parametros(void);
```

```
        void abro_menu_mostrar(void);
        int menu_mostrar(void);
        int raton_menu_mostrar(XButtonEvent *theEvent);
        void cierro_menu_mostrar(void);
        void set_fichero_mostrar(void);

        void muestra_simulacion(void);
        void esquema_mostrar(void);

        void abro_menu_grabacion(void);
        int menu_grabacion(void);
        int raton_menu_grabacion(XButtonEvent *theEvent);
        void set_grabacion(void);
        void creo_fichero(void);
        int raton_set_grabacion(XButtonEvent *theEvent);
        void cierro_menu_grabacion(void);

        void abro_control(void);
        void control(void);
        int  raton_control(XButtonEvent *theEvent);
        int  teclado_control(XKeyEvent *theEvent);
        int  raton_seguir(XButtonEvent *theEvent);
        void cierro_control(void);

        void esquema_1(void);
        void esquema_2(void);
        void esquema_3(void);
        double h, t, t0, tf, save;
        double P_caldera, P_condensador, W_refrigerante, T_refrig_in;
        double T_refrig_out, m_refrigerante, W_bomba, W_valvula1, kx_turbina;
        double h_supercal, m_supercal, k_supercal;

        double num_iter,hit,N_grupos_dat;
        int num_fich, file_mostrar, fh;
        char filename[15];
        FILE *fp;

        double x, y, x1=60., x2=250., y1=20., y2=170;

        void cambio_coordenadas(double u, double v, double u1, double v1)
                { x=x1+u*(x2-x1)/u1;  y=y2-v*(y2-y1)/v1; }
public:
        Scheduler () {}
        void start(void);
};
```

En los apartados correspondientes a las versiones de los programas, podremos especificar de manera más detallada cómo actúa el Scheduler en la gestión de ficheros, menús, ventanas, etc.

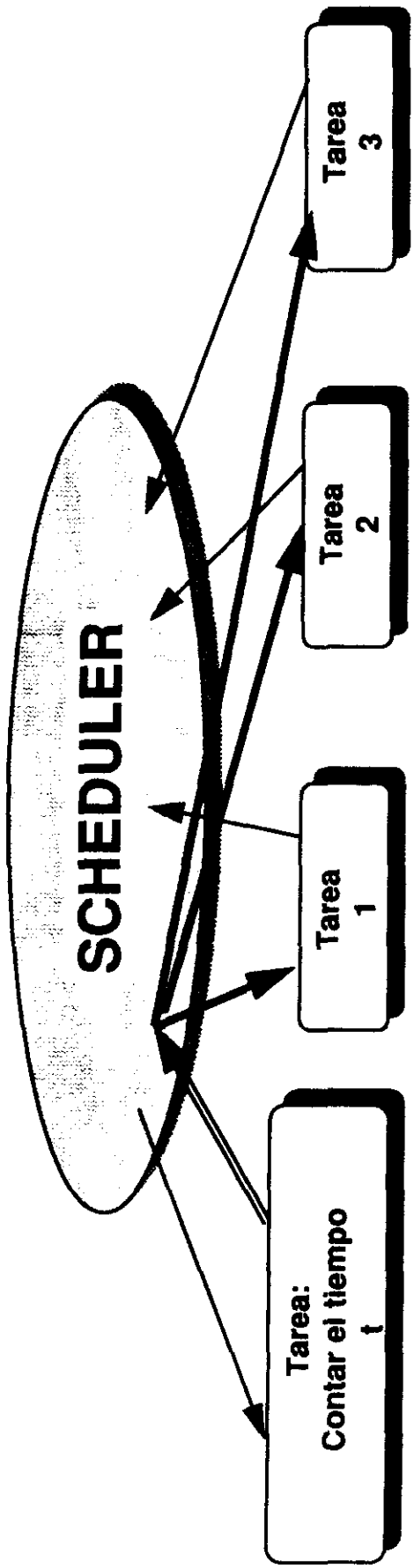


Figura 6.4

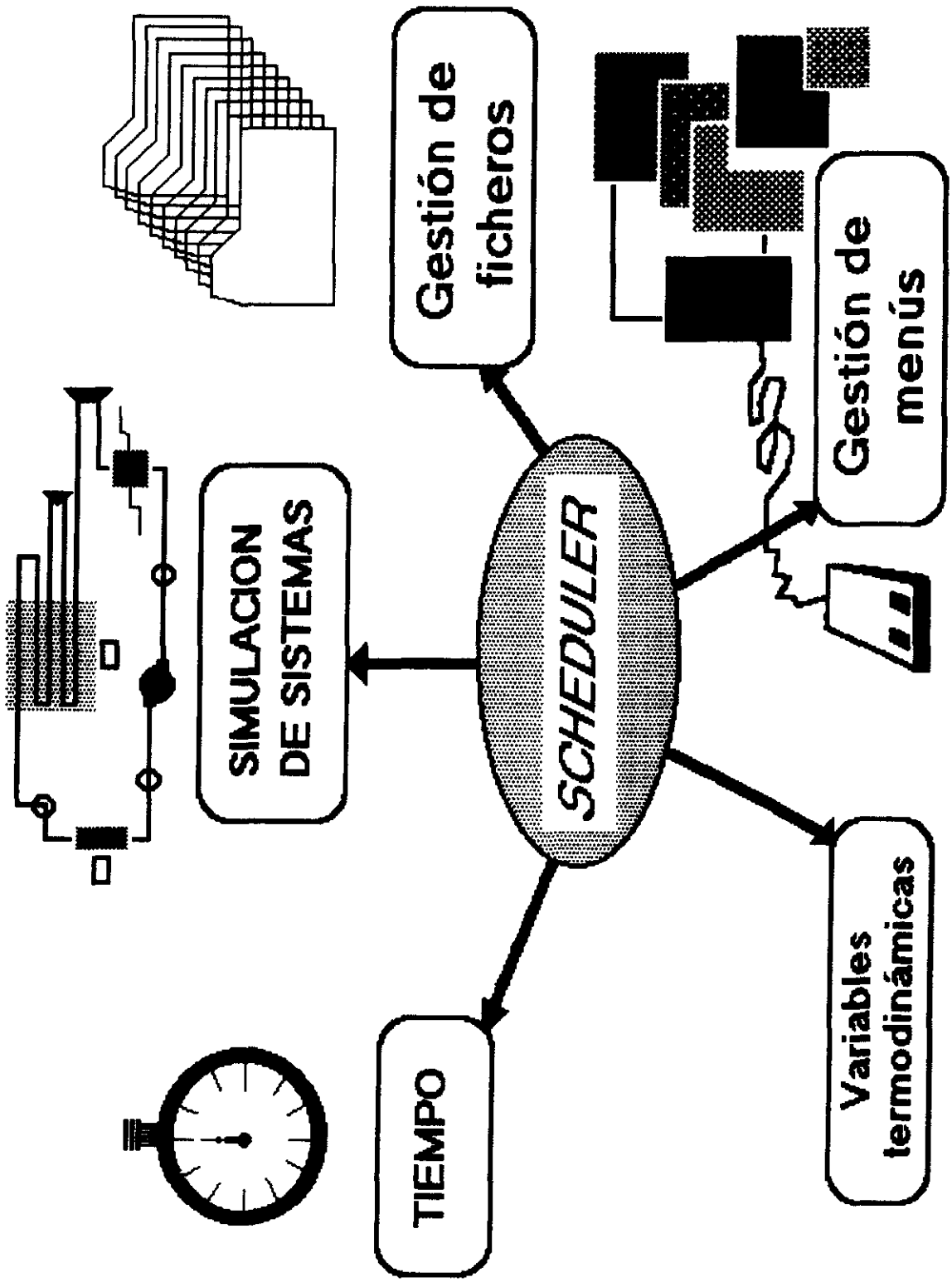

Gestión de ficheros
Gestión de menús
SIMULACION DE SISTEMAS
SCHEDULER
TIEMPO
Variables termodinámicas

Figura 6.5

VI.3.- ASPECTOS GENERALES DE LA PROGRAMACION.-

En este apartado detallaremos las diversas técnicas que hemos empleado para el desarrollo de nuestro software, para pasar de nuestras concepciones al código del programa. Si bien los dos programas creados mantienen ciertas diferencias (pues tienen ámbitos y plataformas distintas), éstas pueden reducirse al terreno gráfico e interactivo, de forma que el resto (la simulación en sí, los cálculos, etc.) es similar. Por ello, algunos de los aspectos de la programación que a continuación serán detallados, son comunes a ambos programas, mientras que otros son específicos de uno u otro.

La finalidad de este apartado es doble. Por una parte, mostrar cómo se pueden implementar en un programa diversas acciones, técnicas, etc; y por otra, facilitar la comprensión del código c++ a aquellos que estén interesados en su estudio.

VI.3.1.- ESTRUCTURAS GENERALES. MENUS.-

Los programas tienen unas estructuras generales, que podemos relacionar con el tronco de un árbol a partir del cual surgen sucesivas ramificaciones. Hemos construido estas estructuras generales paralelamente al soporte de los menús, o dicho de otra forma: la estructura de menús es la estructura general del programa. Así, cada opción desarrolla una "rama" a partir de la cual surgen las diversas funciones, objetos, etc., que elaborarán tareas determinadas.

Cada uno de los programas difieren ligeramente en su estructura general, pues hay que adecuar, en el caso del programa **planta**, la estructura de menús con la técnica que hemos tenido que elaborar para abrir y cerrar ventanas en X-Window.

a) Estructura general del programa PLANTA.EXE.

En la figura 6.6 podemos observar lo que podría ser un diagrama "lógico" para una estructura cualquiera de menús. En un menú 1 podemos hacer una opción (o1) mediante la cual, o pasamos a un menú 2 (o1=1) o finalizamos el programa (o1=0).

Así, en el menú 2 podemos elegir pasar a otro menú (o2=1), o bien retornar al primero (o2=0); por último, suponemos que el último menú, tras realizar las acciones requeridas, sólo puede retornar al anterior menú (o3=0).

Para programar en un "diagrama de bloques" todo ésto, lo primero que se nos ocurriría es algo como el diagrama mostrado en la figura 6.7, en el que se realiza el anterior diagrama lógico. Ahora bien, para evitar el empleo, poco recomendado, de la orden *goto*, remodelamos el diagrama con el fin de usar bucles *do-while*.

En la figura 6.8 podemos ver el diagrama lógico del programa que nos ocupa. En un primer menú elegimos un esquema (un sistema termodinámico); en el segundo, decimos si lo que queremos es simular, o bien mostrar los resultados de alguna simulación que ya se haya realizado sobre el sistema elegido. Según la opción escogida, pasaremos a modificar algún parámetro de inicialización (02=1, menú de variaciones, en el cual elegiremos seguir, o3=1, o volver al menú 2, 03=0) ; o bien a escoger un fichero de datos para recuperar una simulación (o2=2, menú de ficheros, donde mediante la opción o4 elegiremos un fichero, o volver).

En el caso de realizar una simulación, hemos de decidir, mediante un menú con la opción o7, si queremos guardar la simulación en un fichero (o7=1), o no (07=0). Tanto si estamos simulando o recuperando una simulación, el último menú nos permite decidir qué tipo de gráfico (ventana que llena toda la pantalla) deseamos mostrar, pues cada sistema dispone de uno o más gráficos posibles para su visualización.

Por último, en la figura 6.9 podemos ver el diagrama de bloques correspondiente, que será interpretado en lenguaje C.

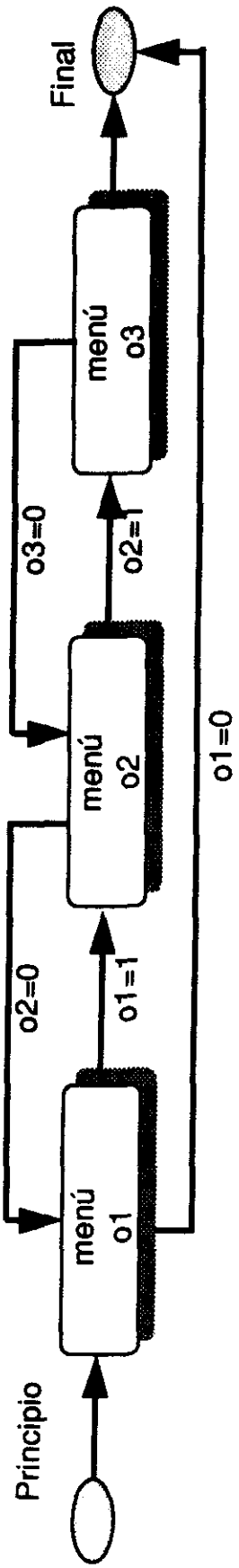


Figura 6.6

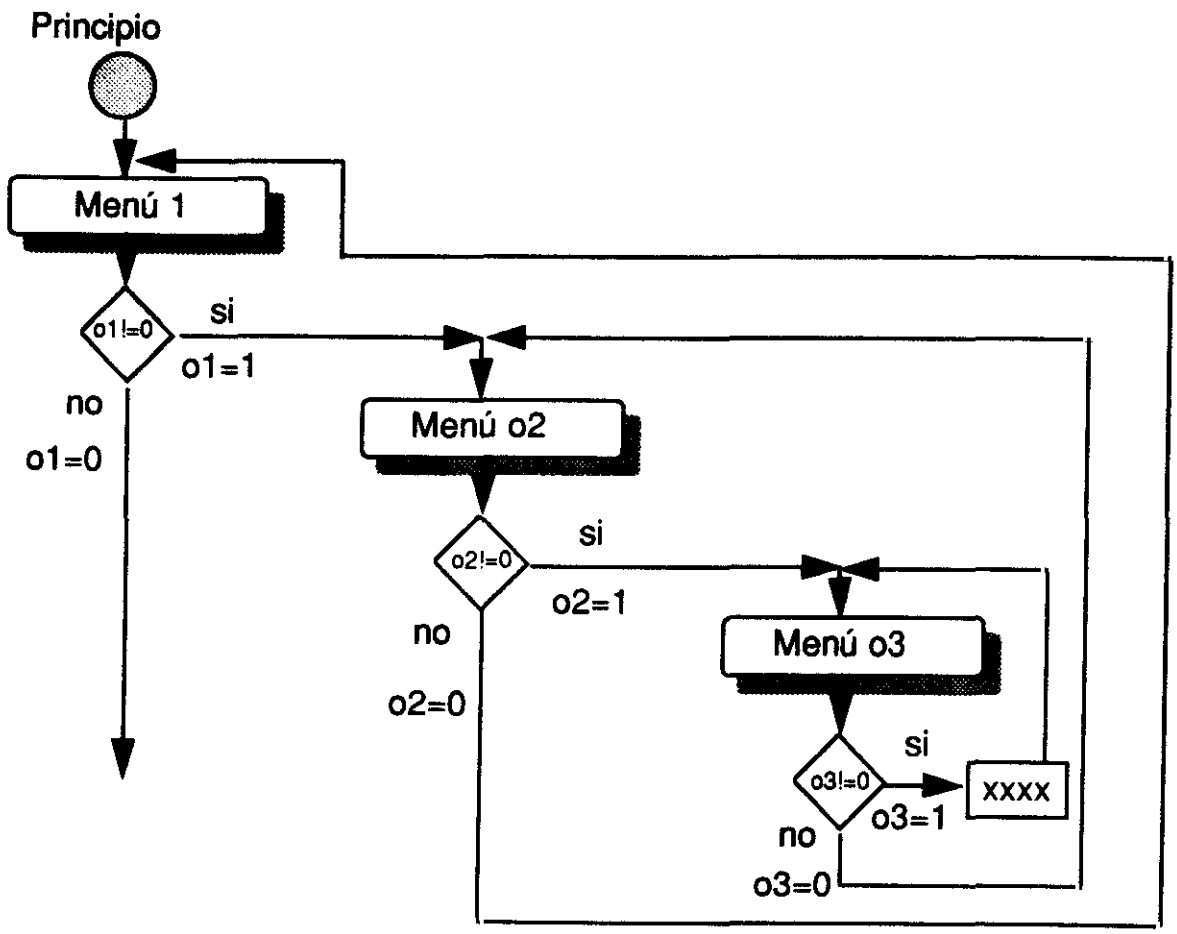

Principio
Menú 1
o1!=0
si
o1=1
no
o1=0
Menú o2
o2!=0
si
o2=1
no
o2=0
Menú o3
o3!=0
si
o3=1
xxxx
no
o3=0

Figura 6.7

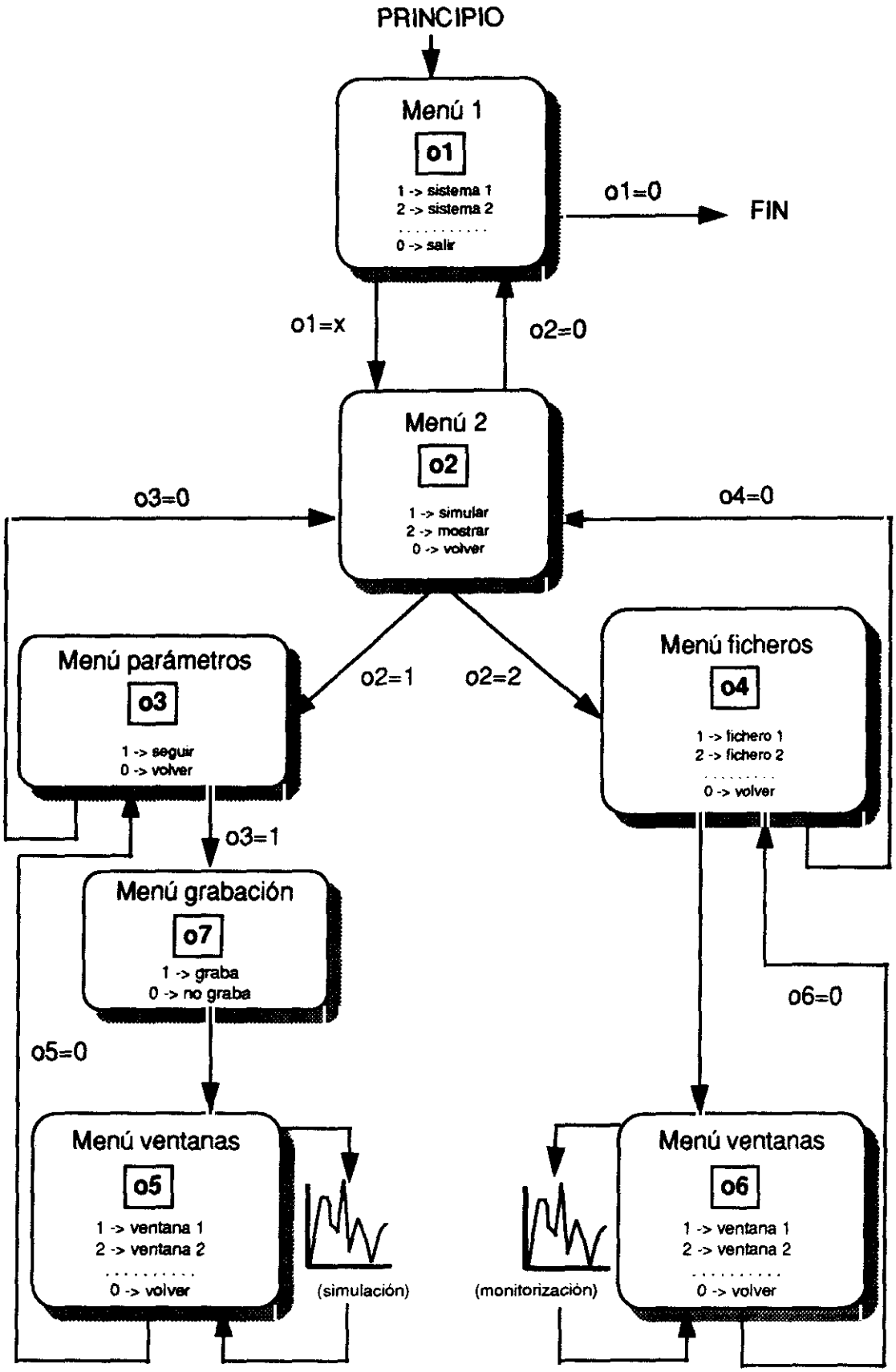

PRINCIPIO
Menú 1
o1
1 -> sistema 1
2 -> sistema 2
0 -> salir
o1=0
FIN
o1=x
o2=0
Menú 2
o2
1 -> simular
2 -> mostrar
0 -> volver
o3=0
o4=0
Menú parámetros
o3
1 -> seguir
0 -> volver
o2=1
o2=2
Menú ficheros
o4
1 -> fichero 1
2 -> fichero 2
0 -> volver
o3=1
Menú grabación
o7
1 -> graba
0 -> no graba
o6=0
o5=0
Menú ventanas
o5
1 -> ventana 1
2 -> ventana 2
0 -> volver
(simulación)
(monitorización)
Menú ventanas
o6
1 -> ventana 1
2 -> ventana 2
0 -> volver

Figura 6.8

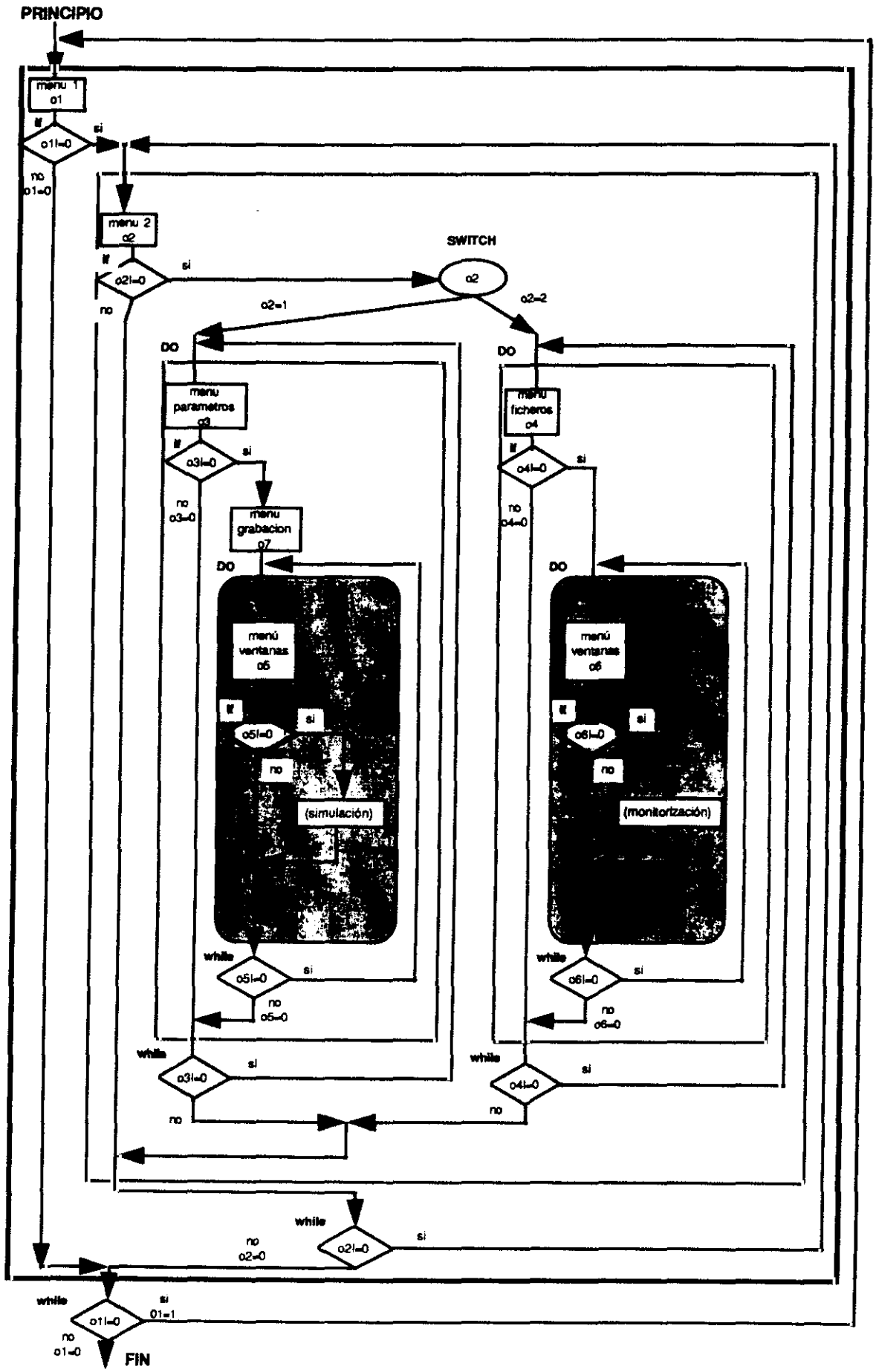

PRINCIPIO
menu 1
o1
if
o1!=0
si
no
o1=0
menu 2
o2
if
o2!=0
si
no
SWITCH
o2
o2=1
o2=2
DO
DO
menu
parametros
o3
if
o3!=0
si
no
o3=0
menu
grabacion
o7
menu
ficheros
o4
if
o4!=0
si
no
o4=0
DO
DO
menú
ventanas
o5
if
o5!=0
si
no
(simulación)
menú
ventanas
o6
if
o6!=0
si
no
(monitorización)
while
o5!=0
si
no
o5=0
while
o6!=0
si
no
o6=0
while
o3!=0
si
no
while
o4!=0
si
no
while
o2!=0
si
no
o2=0
while
o1!=0
si
o1=1
no
o1=0
FIN

Figura 6.9

b) Estructura general del programa *planta*.

En la figura 6.10 podemos observar directamente el diagrama de bloques de la estructura general de menús del programa *planta*. Presenta algunas diferencias respecto del otro programa, pues hemos variado tanto su entorno (X-Window, estaciones de trabajo) como su aspecto general. A mayor cantidad de gráficas, más lentitud de simulación. Esta lentitud se deja sentir en los PC, de modo que es necesario restringir la visualización, eligiendo qué gráficos deseamos llevar a pantalla. Ahora bien, la mayor potencia de proceso de las estaciones de trabajo minimiza el problema, hasta el punto que nos podemos permitir el lujo de incorporar todas las gráficas juntas, por lo que el menú de elección de gráficas deja de existir.

Por otro lado, y debido a la incorporación de interactividad por ratón y teclado tanto en los menús como en la variación de parámetros en tiempo de simulación, se crearán menús nuevos (como el de control) y numerosos pequeños menús (botones) para desplegar ayudas, confirmar parámetros, etc.

La estructura general es, en grandes líneas, similar a la del anterior programa. Obviaremos la densa descripción de los valores que han de tener los distintos parámetros de opciones (o1, o2, ...) para realizar las tareas que a continuación se relatan (todo ello puede verse en el listado del programa proporcionado en el apéndice).

Así, en un menú inicial, elegimos el sistema, la ayuda general, o la finalización del programa. Caso de elegir un sistema, se abrirá un menú en el cual elegiremos el modo de ejecución, esto es, simular o mostrar, o bien pedir una explicación o retornar al primer menú. En el caso de simular, se abre el menú de parámetros de inicialización, con la opción de continuar o volver; si continuamos, se nos pedirá si queremos almacenar la simulación en un fichero o no, volver o desplegar ayudas; caso de continuar, se abre el menú de control, donde podremos iniciar, parar, continuar o cancelar la simulación. En el caso de mostrar una simulación guardada en un fichero de datos, tan sólo necesitamos un menú de elección de fichero, en el cual elegiremos, entre una lista de ficheros que se nos informará de su disponibilidad, el que nos convenga.

Todo esto, a la vista de un usuario del programa, quedará visualizado mediante multitud de botones de "O.k.", "cancelar", "volver", "parar", "continuar", "incrementar", "ayuda", "explicación", "abrir", "cerrar", etc.

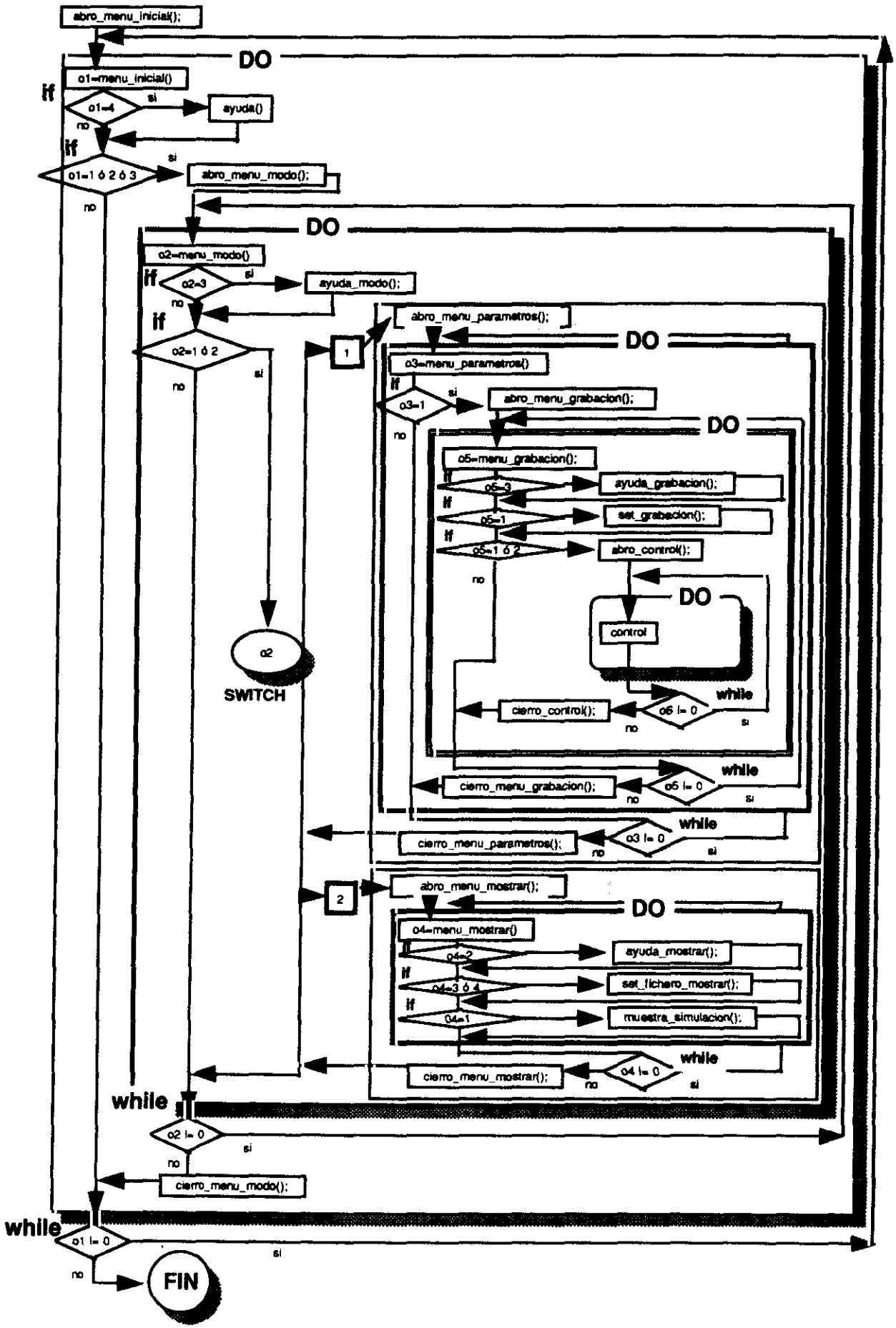

abro_menu_inicial();
DO
o1=menu_inicial()
if
o1=4
si
ayuda()
no
if
si
o1=1 ó 2 ó 3
abro_menu_modo();
no
DO
o2=menu_modo()
if
o2=3
si
ayuda_modo();
no
if
o2=1 ó 2
si
no
1
abro_menu_parametros();
DO
o3=menu_parametros()
if
o3=1
si
abro_menu_grabacion();
no
DO
o5=menu_grabacion();
if
o5=3
ayuda_grabacion();
if
o5=1
set_grabacion();
if
o5=1 ó 2
abro_control();
no
DO
control
while
o6 != 0
si
no
cierro_control();
o2
SWITCH
while
o6 != 0
si
no
cierro_menu_grabacion();
while
o3 != 0
si
no
cierro_menu_parametros();
2
abro_menu_mostrar();
DO
o4=menu_mostrar()
o4=2
ayuda_mostrar();
if
o4=3 ó 4
set_fichero_mostrar();
if
o4=1
muestra_simulacion();
while
o4 != 0
si
no
cierro_menu_mostrar();
while
o2 != 0
si
no
cierro_menu_modo();
while
o1 != 0
si
no
FIN

Figura 6.10

VI.3.2.- GESTION DE FICHEROS. TOMA Y LECTURA DE DATOS.-

El Scheduler lleva directamente el control de ficheros en cuanto a su creación, apertura, escritura, lectura, etc. La simulación genera datos cada h segundos, desde t0 hasta tf. Cuando nos interese, guardaremos diversos datos de la simulación en un fichero, para poder reconstruir la simulación efectuada cuando queramos.

La ventaja de utilizar estos ficheros de simulaciones es que, cuando reconstruimos una simulación mediante la lectura de datos del fichero, éstos no se calculan -no se resuelven ecuaciones- sino que se leen, convenientemente dispuestos y organizados, de manera que la visualización de la simulación es mucho más rápida pues el procesador del ordenador lee datos a más velocidad que si tuviera que recurrir a todo tipo de operaciones aritmético-lógicas. Por otro lado, podemos estudiar más detenidamente las evoluciones de los procesos termodinámicos, comparando diversas simulaciones de una forma rápida. También tiene un pequeño inconveniente: si efectuamos una simulación que va guardando datos en un fichero, consumirá más tiempo que si no lo hiciese.

Podemos tomar todos los datos que una simulación genera, pero ello sería desaconsejable cuando su número, como es nuestro caso, es muy elevado, ya que mermaría la rapidez de la simulación, sobre todo cuando h fuese pequeño y tf grande. También, no tendría sentido almacenar datos que, o bien no vamos a necesitar, o no vamos a representar gráficamente para su estudio. Por tanto, vamos a guardar en los ficheros una serie de variables (distinta para cada sistema, por supuesto), agrupadas en cada toma de datos, de forma que tengamos un número total de **N_grupos_dat** tomas de datos (en cada toma, un cierto número de variables); esto se hará cada **hit** iteraciones de la simulación, sabiendo que cada iteración se produce cada h segundos. Emplearemos las siguientes variables:

t → tiempo.

t0 → instante inicial (segundos).

tf → instante final (segundos).

h → paso (segundos). Intervalo de generación de datos en la simulación.

num_iter → nº total de iteraciones o cálculos de los datos generados.

$$[\ tf = t0 + num\_iter * h\ ]$$

hit → nº de iteraciones que dejamos pasar entre cada toma de datos.

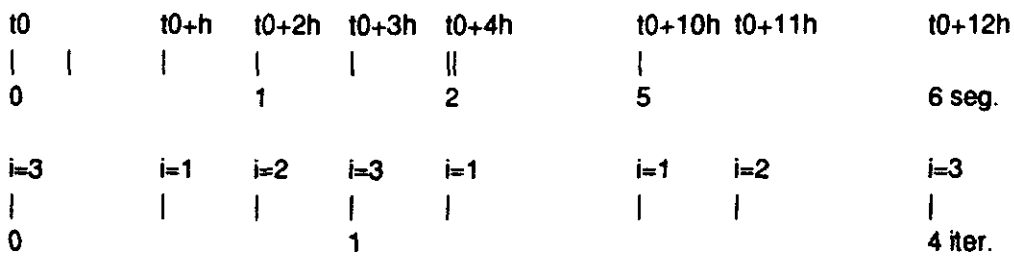
N_grupos_dat → nº total de grupos de datos generados guardados en cada toma.

$$[\ N\_dat = \frac{num\_iter}{hit}\ ]$$

La estructura básica de las iteraciones de la simulación es así:

```
for(cont=hit , t=t0 ; t<tf ; t=t+h , cont++)
        {
        .....
        ..... calculamos las variables en t
        .....
if(cont==hit)
        {
        ..... escribimos el dato (t) en un fichero
        cont=0;
        }
}
```

Por poner un ejemplo, si t0=0 s, tf=6 s y h=0.5 s, entonces num_iter=12. Si decidimos que hit=3 (esto es, cada tres iteraciones de la simulación guardamos datos), entonces N_grupos_dat=4 (o sea, cada guardamos en el fichero un total de 4 tiempos de simulación, en cada uno de los cuales están las variables termodinámicas consideradas).



A continuación, y para mostrar el efecto que suponen las distintas posibilidades en la simulación en cuanto a consumo de tiempo, proponemos el siguiente cuadro, tomado en una simulación sobre una plataforma Sun-Sparc 1+ :

t0=0 tf=3000 num_iter=	SIMULACION	SIMULACION (grabando datos)	SIMULACION (leyendo datos)
hit=10 N_grupos_dat=600	108 seg	120 seg (+ 10 %)	12 seg (- 800 %)
hit=30 N_grupos_dat=200	108 seg	116 seg (+ 6.8 %)	5 seg (- 2060 %)
hit=100 N_grupos_dat=60	108 seg	114 seg (+ 5.2 %)	2 seg (- 5300 %)

VI.3.3.- MANEJO DE VENTANAS.-

En la programación de nuestras aplicaciones es sumamente aconsejable el uso de un contexto gráfico que nos permita abrir y cerrar ventanas con capacidad de interactuar con el ratón y el teclado, debido a las numerosas distintas acciones que han de considerarse, y que van desde la pura visualización de una magnitud a través del tiempo hasta la variación de parámetros, pasando por la petición de ayudas, recuperación de ficheros, selección de modos, etc.

Esto se puede conseguir mediante entornos gráficos "windows", como son Microsoft Windows (para PC's) y MIT X-Windows (para estaciones de trabajo). Como ya hemos mencionado en otros momentos, nuestra aplicación encuentra su lugar idóneo de trabajo en una workstation, entre otras razones porque la simulación va a necesitar un ordenador potente en cálculo y veloz en procesamiento; pero también por el soporte gráfico que X-Window ofrece. Es por ello, que la aplicación completa en sus aspectos de presentación, interactividad, etc. ha sido la versión *planta* para estaciones de trabajo.

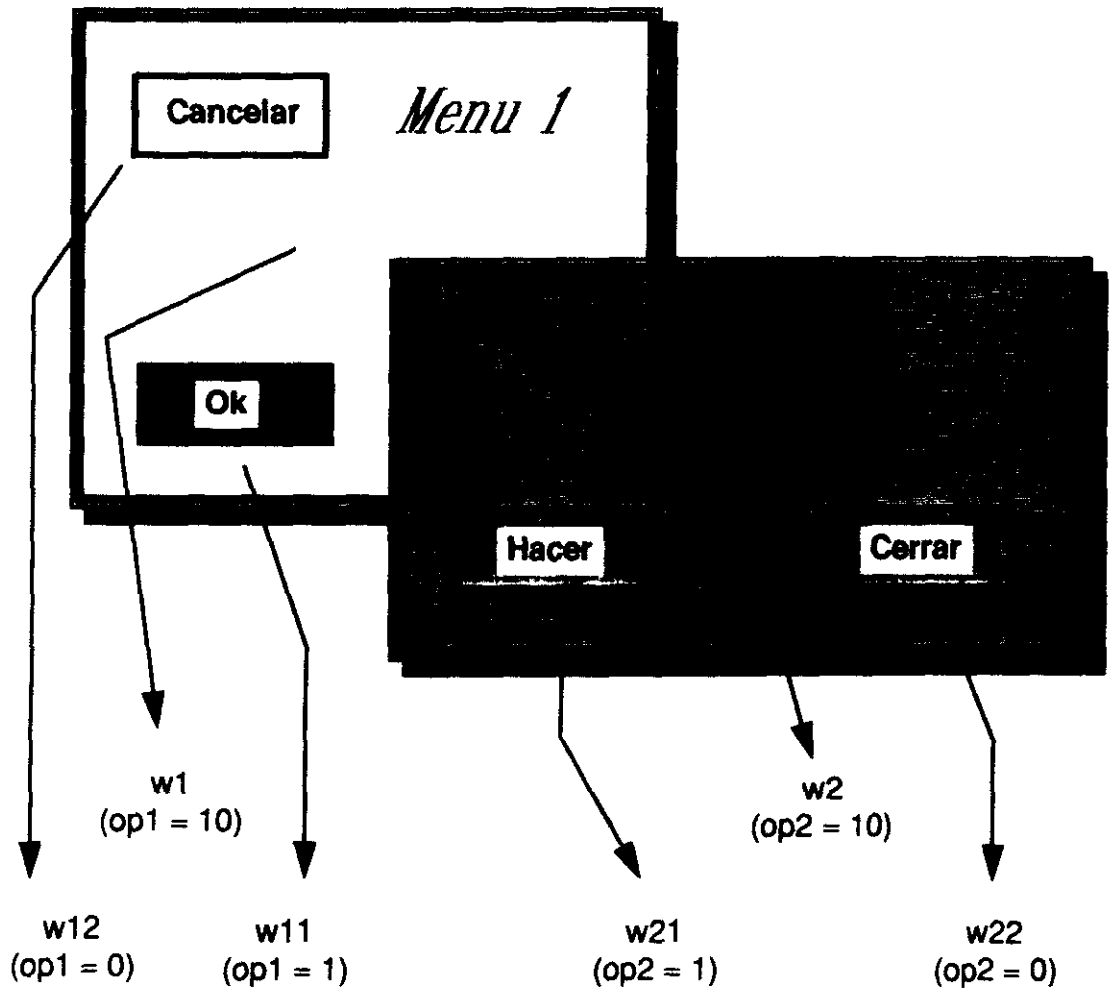
En nuestro uso de X-Window no hemos podido ayudarnos de utilidades de generación de ventanas, de forma que hemos tenido que construirlas programando directamente todas nuestras necesidades, ayudados por la biblioteca Xlib, que es rápida de ejecución.

A continuación ofrecemos un pequeño y muy sencillo ejemplo de cómo hemos organizado la forma de manejar las ventanas con los eventos. Puede ser útil estudiar su código C++ (ver apéndice, listado planta).

Así, y según se ve en la figura 6.11, supongamos que hemos abierto, mediante la rutina "abro_menu_1()", una ventana principal w1 y dos botones (las también ventanas w11 y w12, sub-ventanas de w1): "Ok" y "Cancelar". Entrando en un ciclo do-while, mediante la función "int menu_1()" damos un valor al parámetro op1; queremos que, si el ratón se acciona sobre la ventana w11, op1 valga 1, sobre w12, 0, y en cualquier otra ventana, op1=10. Así, si seleccionamos el ratón sobre "Ok", vamos a la rutina "abro_menu_2()", que abre las ventanas w2, w21 y w22. Aquí repetiríamos de una forma similar las acciones de elegir un valor para el parámetro op2 mediante "int menu_2()" (de forma que si el ratón se presiona sobre la ventana w21, op2=1, sobre w22 op2=0, y sobre cualquier otra ventana - w2, w1, w11, etc.- , op2 valga 10). Si op2 toma el valor 1 (actuándo sobre la ventana "Hacer"), se efectúa una tarea cualquiera, volviendo al ciclo do-while del menú 2; si presionamos sobre cualquier otra ventana, continuamos sin salir de dicho ciclo; pero si op2 toma el valor 0 (presionando sobre

"Cerrar"), pasaríamos a cerrar las ventanas w2, w21 y w22., entrando de nuevo en el ciclo do-while correspondiente al menú 1. De aquí no podríamos salir a menos que diéramos el valor 0 a op1, en cuyo caso cerraríamos las ventanas w11, w12 y w1.

En los listados de la figura 6.12 podemos ver desarrolladas las rutinas que nos permiten efectuar todo lo anteriormente dicho.




```
abro_menu_1()                -> creo las ventanas w1, w11 y w12.
do {                          -> inicio del bucle para el menú 1
   op1=menu_1();             -> escojo opción
   if(op1==1)                 -> si op1=1, abriré el segundo menú
      {
      abro_menu_2();       -> abro las ventanas w2, w21 y w22
      do {                   -> inicio del bucle para el menú 2
         op2=menu_2(); -> escojo opción
         if(op2==1)         -> si op2=2, efectúo una tarea determinada
            { . . . . . }
         }
         while(op2!=0);   -> salgo del menú 2 cuando op2 sea 0
      cierro_menu_2();     -> cierro las ventanas w2,1 w22 y w2
      }
   }
   while(op1!=0);            -> salgo del menú 1 cuando op1 sea 0
cierro_menu_1();             -> cierro las ventanas w11, w12 y w1
```

Figura 6.11

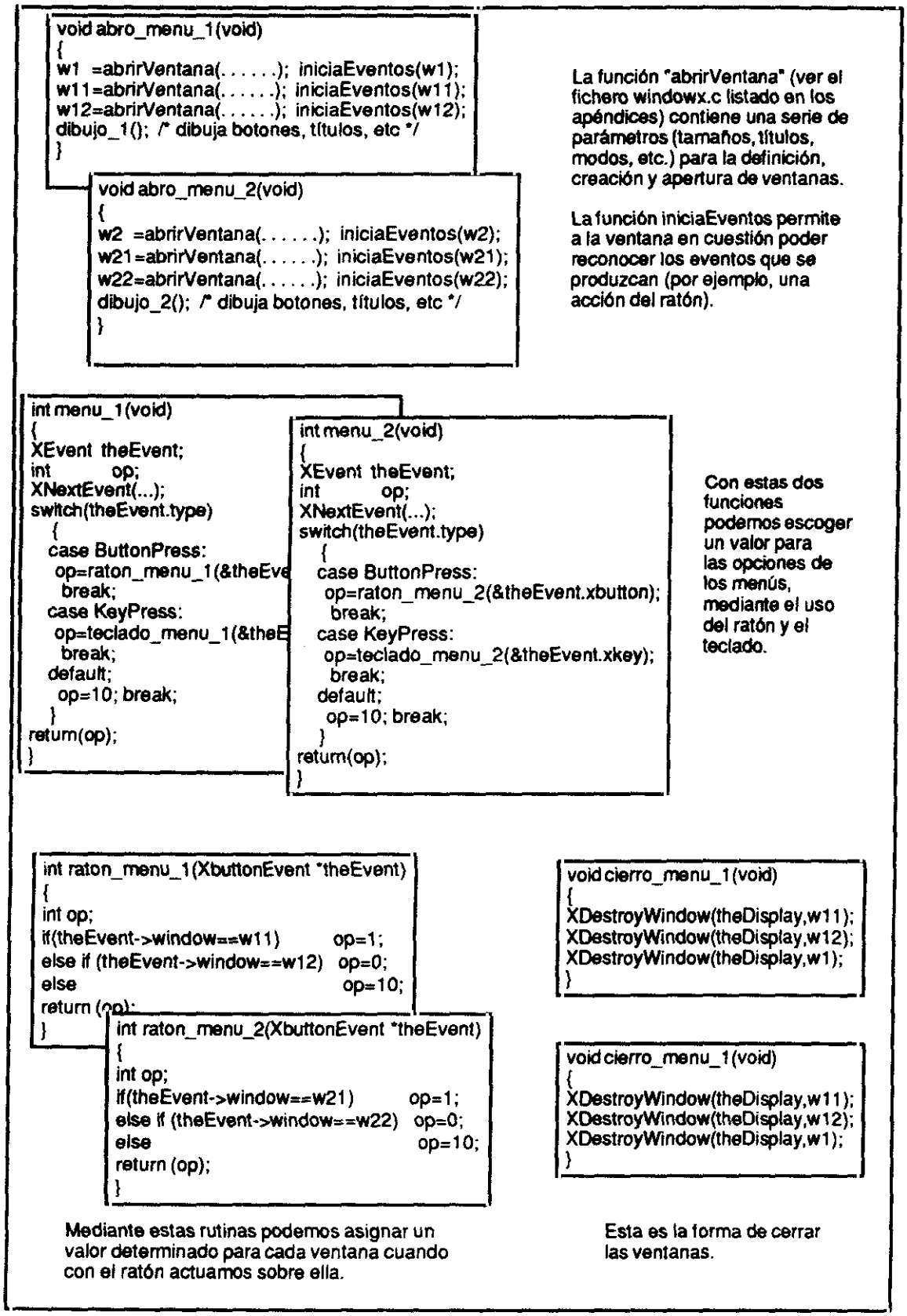

void abro_menu_1(void)
{
w1 =abrirVentana(. . . . .); iniciaEventos(w1);
w11=abrirVentana(. . . . .); iniciaEventos(w11);
w12=abrirVentana(. . . . .); iniciaEventos(w12);
dibujo_1(); /* dibuja botones, títulos, etc */
}
void abro_menu_2(void)
{
w2 =abrirVentana(. . . . .); iniciaEventos(w2);
w21=abrirVentana(. . . . .); iniciaEventos(w21);
w22=abrirVentana(. . . . .); iniciaEventos(w22);
dibujo_2(); /* dibuja botones, títulos, etc */
}
La función "abrirVentana" (ver el fichero windowx.c listado en los apéndices) contiene una serie de parámetros (tamaños, títulos, modos, etc.) para la definición, creación y apertura de ventanas.
La función iniciaEventos permite a la ventana en cuestión poder reconocer los eventos que se produzcan (por ejemplo, una acción del ratón).
int menu_1(void)
{
XEvent theEvent;
int op;
XNextEvent(...);
switch(theEvent.type)
{
case ButtonPress:
op=raton_menu_1(&theEve
break;
case KeyPress:
op=teclado_menu_1(&theE
break;
default;
op=10; break;
}
return(op);
}
int menu_2(void)
{
XEvent theEvent;
int op;
XNextEvent(...);
switch(theEvent.type)
{
case ButtonPress:
op=raton_menu_2(&theEvent.xbutton);
break;
case KeyPress:
op=teclado_menu_2(&theEvent.xkey);
break;
default;
op=10; break;
}
return(op);
}
Con estas dos funciones podemos escoger un valor para las opciones de los menús, mediante el uso del ratón y el teclado.
int raton_menu_1(XbuttonEvent *theEvent)
{
int op;
if(theEvent->window==w11) op=1;
else if (theEvent->window==w12) op=0;
else op=10;
return (op);
}
int raton_menu_2(XbuttonEvent *theEvent)
{
int op;
if(theEvent->window==w21) op=1;
else if (theEvent->window==w22) op=0;
else op=10;
return (op);
}
Mediante estas rutinas podemos asignar un valor determinado para cada ventana cuando con el ratón actuamos sobre ella.
void cierro_menu_1(void)
{
XDestroyWindow(theDisplay,w11);
XDestroyWindow(theDisplay,w12);
XDestroyWindow(theDisplay,w1);
}
void cierro_menu_1(void)
{
XDestroyWindow(theDisplay,w11);
XDestroyWindow(theDisplay,w12);
XDestroyWindow(theDisplay,w1);
}
Esta es la forma de cerrar las ventanas.

Figura 6.12

VI.3.4.- ESTRUCTURAS DE SIMULACION DE LOS SISTEMAS.-

En el primer apartado del presente Capítulo exponíamos varios ejemplos de sistemas a simular, y la forma de resolver las ecuaciones en el tiempo para efectuar el estudio dinámico. Lo que nos proponemos en este apartado es describir la estructura de la que nos hemos servido para implementar, en código C++, la simulación dinámica de los sistemas.

Hay pequeños detalles que varían en la aplicación realizada para PC y la realizada para la estación de trabajo; ello radica en el hecho de que la presencia de X Windows en la última introduce distintas formas de interactuar con el teclado y el ratón, junto con una expresión gráfica distinta. Pero, en general, podemos considerar nuestra estructura como la dispuesta en la figura 6.13. Allí podemos ver una serie de tareas que se realizan en cierto orden; las que aparecen en recuadros más gruesos son prácticamente comunes en las aplicaciones *PLANTA.EXE* y *planta*, mientras que las demás tienen sus variaciones.

En general, la estructura responde al siguiente esquema: En primer lugar, declaramos las necesarias variables de trabajo. A continuación, declaramos e inicializamos los objetos que representarán los distintos dispositivos del sistema, con una serie de parámetros inicializados para el instante inicial. Posteriormente comprobamos si se pidió guardar la simulación en un fichero; en caso afirmativo, abrimos y preparamos un fichero el nombre y características que se dieron en el menú correspondiente. Tras abrir el sistema gráfico correspondiente, se inicia el ciclo de cálculo y representación, que es la simulación en sí; de esta forma, a partir de unos valores iniciales (valores en t), se asignan y calculan los valores de otras variables en dicho instante (mediante relaciones lineales -ecuaciones no diferenciales, que suponen una relación instantánea-, y asignaciones tales como que la temperatura de salida de un elemento en el instante t es la temperatura de entrada del siguiente dispositivo en el mismo instante). Chequeando si se pidió guardar la simulación en un fichero, se almacenarán en éste determinados valores. En todo caso, a continuación se dibujan los puntos (que representan los valores de distintas variables) sobre las gráficas. Antes de cerrar el ciclo, hemos de realizar dos importantes tareas: primero, variar los parámetros de control si así se pidió, de forma que el nuevo valor será tomado como el que posea en el siguiente instante, t+h; segundo, calcular los valores de ciertos parámetros mediante la resolución de las ecuaciones diferenciales (o sea, conociendo el valor de la variable x en t, hallamos su valor en t+h, resolviendo la ecuación dx/dt=. . .); el resultado de estas dos tareas nos permiten inicializar el sistema para el siguiente instante t+h, en el cual se repite el ciclo. Por último, cerramos los sistemas gráficos y ficheros correspondientes.

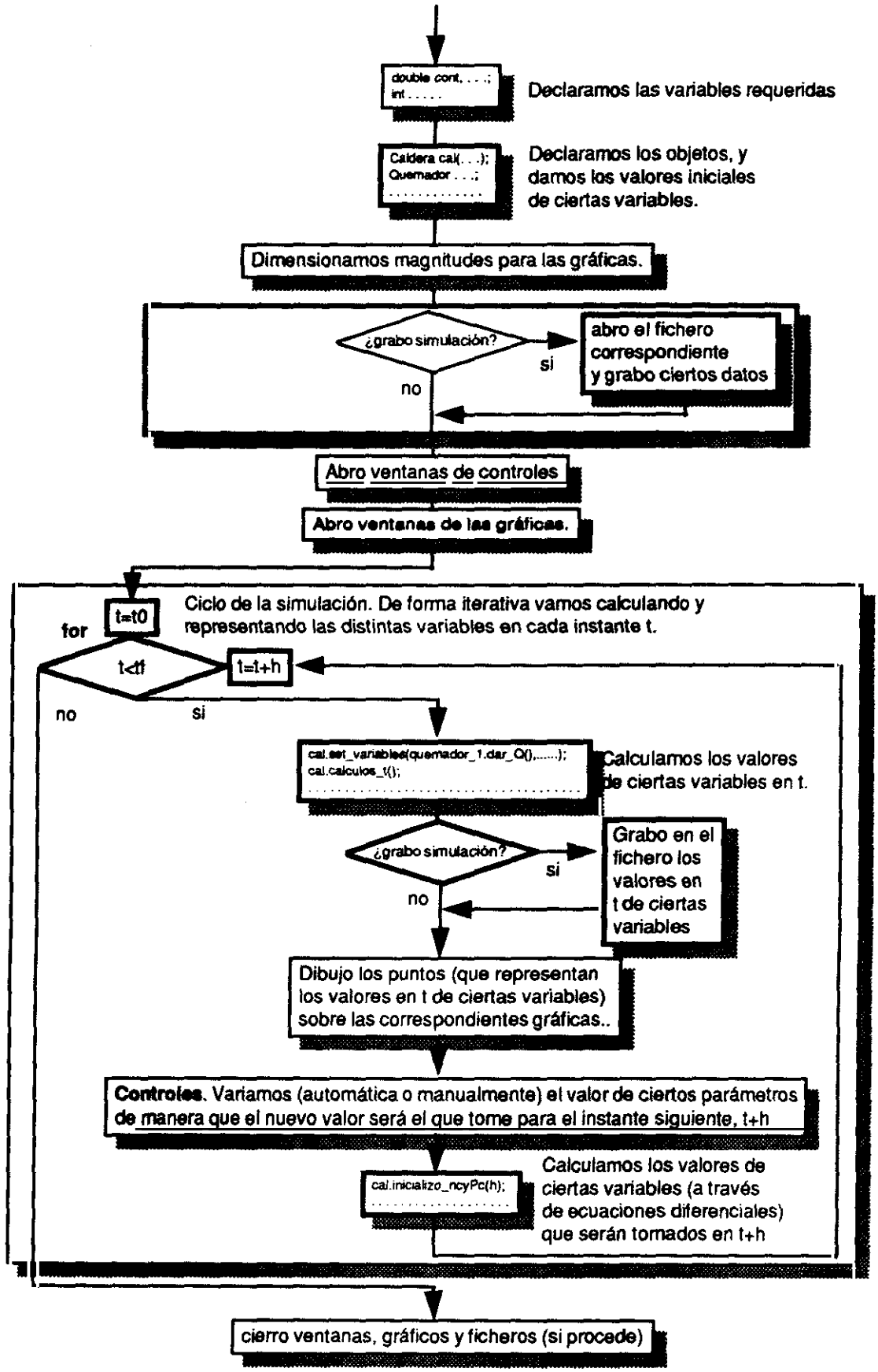

double cont, . . .;
int . . . . .
Declaramos las variables requeridas
Caldera cal(. . .);
Quemador . . .;
Declaramos los objetos, y damos los valores iniciales de ciertas variables.
Dimensionamos magnitudes para las gráficas.
¿grabo simulación?
si
no
abro el fichero correspondiente y grabo ciertos datos
Abro ventanas de controles
Abro ventanas de las gráficas.
for
t=t0
t<tf
t=t+h
no
si
Ciclo de la simulación. De forma iterativa vamos calculando y representando las distintas variables en cada instante t.
cal.set_variables(quemador_1.dar_Q(),.....);
cal.calculos_t();
Calculamos los valores de ciertas variables en t.
¿grabo simulación?
si
no
Grabo en el fichero los valores en t de ciertas variables
Dibujo los puntos (que representan los valores en t de ciertas variables) sobre las correspondientes gráficas..
Controles. Variamos (automática o manualmente) el valor de ciertos parámetros de manera que el nuevo valor será el que tome para el instante siguiente, t+h
cal.inicializo_ncyPc(h);
Calculamos los valores de ciertas variables (a través de ecuaciones diferenciales) que serán tomados en t+h
cierro ventanas, gráficos y ficheros (si procede)

Figura 6.13

Los códigos de estas estructuras pueden encontrarse en los correspondientes listados de los apéndices; están implementados como métodos protegidos del Scheduler, con los identificadores "esquema_1(...)", etc.

VI.3.5.- CONTROLES.-

Podemos distinguir dos tipos de control dentro del código C++ de las aplicaciones, y que podríamos denominar de la siguiente manera :

a) Control interactivo en tiempo de simulación.

Se trata de tener un conjunto de variables de estado cuyos valores puedan ser alterados manualmente durante el desarrollo de la simulación; el objeto de ésto es el de dotar de mayor potencialidad a la simulación, permitiendo un amplio grado de experimentación con nuestro sistema; así, podremos observar los cambios ante ciertos comportamientos, simular averías, corregir los valores de variables que se disparan mediante la compensación con otras, suavizar comportamientos, etc.

Gracias a ello, en el apartado VI.6 referido a los resultados de la simulación, podremos enumerar una serie de eventos posibles para el estudio de estos sistemas, y se verá cómo afectan a la marcha de los mismos.

En los apartados VI.4 y VI.5 (descripciones de los programas desarrollados) se encuentran los parámetros que pueden controlarse manualmente en tiempo de simulación, y que varían para cada sistema y para cada aplicación desarrollada.

Obviamente, no se incluye en este tipo de control la posibilidad que tiene nuestra aplicación de cambiar los valores iniciales de varios parámetros, pues si bien ello es interactivo y permite una mayor versatilidad de nuestras simulaciones, no son en tiempo de simulación.

La forma de programar este tipo de control varía en ambas aplicaciones :

— *PLANTA.EXE*:

Dentro del bucle de simulación (ver figura 6.13) colocamos una instrucción lógica **if** que comprueba, mediante la función de librería C **kbhit()** si alguna tecla ha sido presionada; en caso afirmativo, mediante un **switch** discriminamos cuál de los caracteres de teclado que efectúan el control ha sido el presionado, y actuamos en consecuencia sobre la variable correspondiente; el código sería como lo siguiente:

```
if(kbhit()!=0)
    {
    switch(getch())
        {
        case q_KEY: quemador_1.mas_Q(); break;
        case a_KEY: quemador_1.menos_Q(); break;
        .....
        case Esc_KEY: t=tf; break;
        default: break;
        }
    }
```

Los métodos que controlan las variables (incrementando o decrementando en una cantidad determinada) son públicos de las clases a las que correspondan dichas variables; por ejemplo, si quiero aumentar Qes, he de hacer una llamada al siguiente método público de la clase Quemador :

```
void Quemador :: mas_Q(void)
        { Q=Q+1.; }
```

— *planta :*

El empleo de X-Windows nos lleva a otro planteamiento; previamente al ciclo de la simulación, se abren unas ventanas (botones) que representan los controles (botones de incremento y decremento para cada variable de control); una vez iniciada la simulación (dentro del bucle), se comprueba (mediante **XCheckWindowEvent**) si el cursor del ratón ha sido presionado sobre una de estas ventanas, actuándose en consecuencia:

```
if( XCheckWindowEvent( theDisplay, w, EV_MASK, &Event ) == 1 )
    { quemador_1.menos_Q(); }
```

En este ejemplo, w sería la ventana del botón de decremento para Qec.

b) Control automático de variables de estado.

Los valores de diversas magnitudes físicas (aportes energéticos, caudales, masas, etc.) son dados inicialmente de una forma provisional, referencial; a la vista del comportamiento de la simulación (viendo qué variables se disparan, o se mueven en unos niveles no satisfactorios, etc.), podremos deducir de una forma aproximada a qué valores iniciales fijados se deben dichos comportamientos. También es posible que, durante la simulación, la evolución de una de las variables repercuta de manera negativa sobre otras. Otro caso podría ser que al evolucionar de cierta forma una variable, pueda alcanzar unos valores que se saldrían de los recogidos en nuestras tablas termodinámicas, con la consiguiente contaminación de cálculo sobre otras variables (este es el caso más frecuente). Para ayudar a paliar algunos de estos sucesos, recurrimos al control automático.

Este es un tipo de control distinto, pues el operario no interviene en él. Se trata de que el sistema tome por sí mismo unas acciones determinadas ante unos comportamientos dados; podemos con ello dar solución a un buen número de estos sucesos que distorsionan la simulación del sistema.

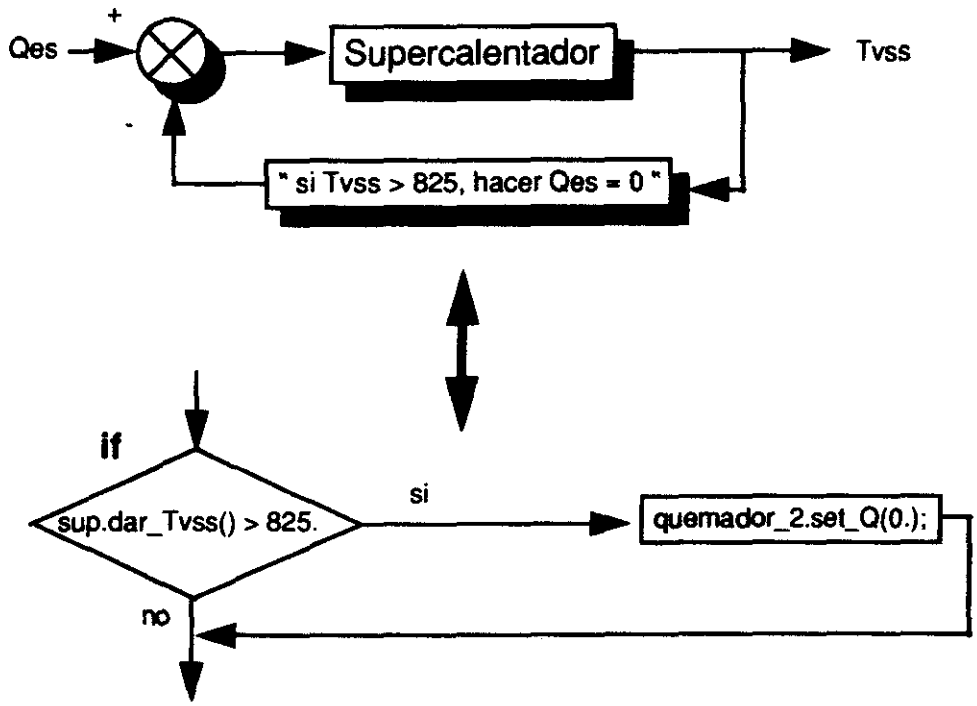
El control automático se implementa en el código del programa dentro del bucle de simulación (ver figura 6.13) en forma de condiciones lógicas. Por ejemplo, hemos observado que si la temperatura del gas de salida del supercalentador, T_{vss}, es superior a 825 ºC, se producen desajustes en los cálculos (pues algunas variables trabajarán sobre zonas que no están tabuladas); una forma de solventar este inconveniente sería cerrar automáticamente el aporte energético al supercalentador (Q_{es}=0), de forma que el vapor de salida estaría a una temperatura cada vez menor (ver figura 6.14).

La adición de estos elementos de control automático en el programa se produce a medida que vamos descubriendo sucesos o anomalías en las simulaciones. Cuanto más estudio y más experimentación dediquemos al sistema, más comprenderemos sobre los niveles y comportamientos de las variables, y por tanto podremos diseñar más mecanismos de control automático que nos eviten ciertos eventos.

VI.3.6.- GRAFICOS.-

La evolución de las diversas magnitudes de interés nos proporciona una gran información sobre la física de los procesos, de modo que interesa enormemente facilitar su monitorización. Para ello, hemos de contar con entornos gráficos para la programación. Nosotros hemos empleado:

- Borland Graphic Interface (BGI). Para la aplicación PLANTA.EXE.
- X Window v11 R4. Para la aplicación planta.


Qes
+
-
Supercalentador
Tvss
" si Tvss > 825, hacer Qes = 0 "
if
sup.dar_Tvss() > 825.
si
quemador_2.set_Q(0.);
no

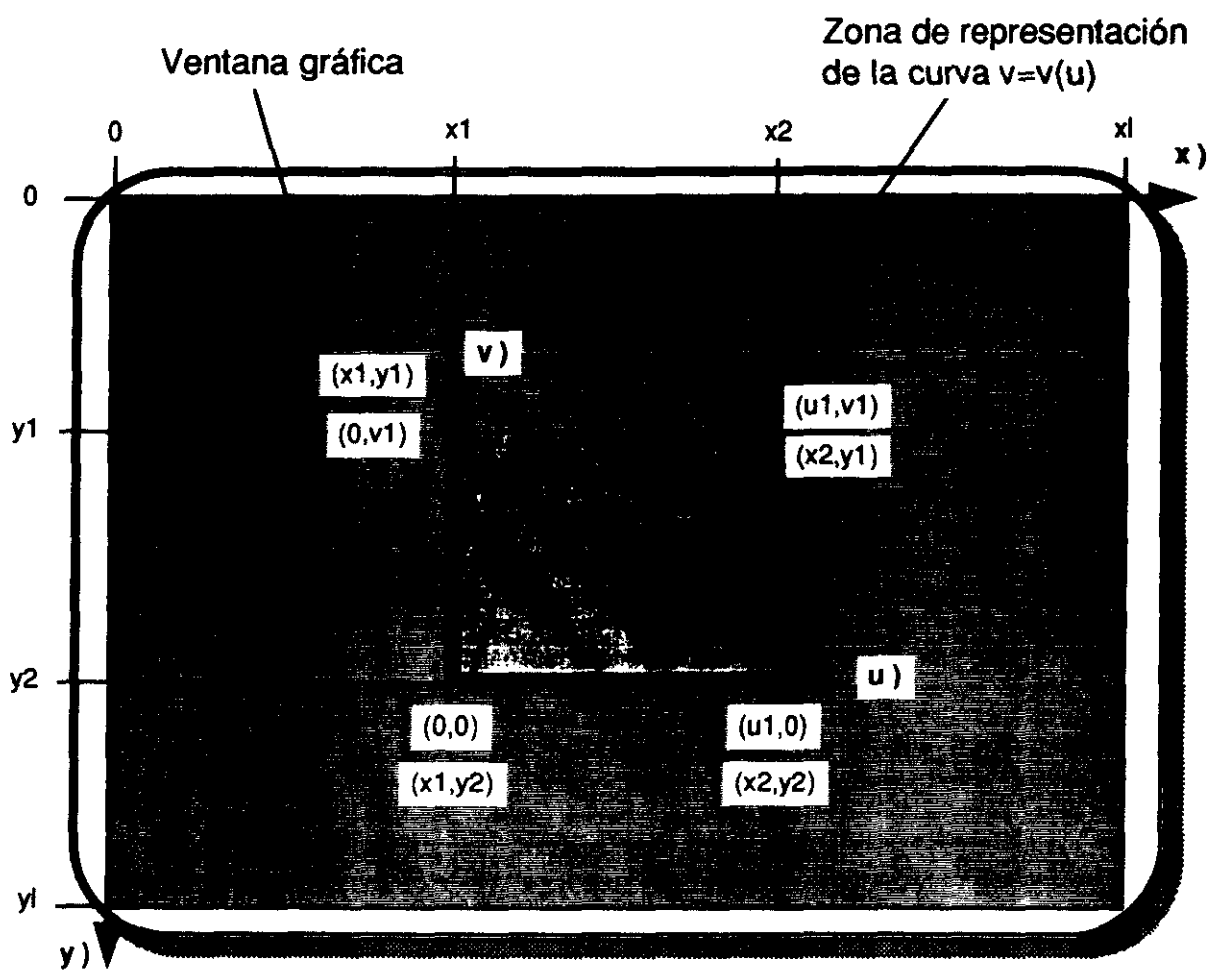

Ventana gráfica
Zona de representación
de la curva v=v(u)
0
x1
x2
xl
x)
0
y1
y2
yl
y)
v)
(x1,y1)
(0,v1)
(u1,v1)
(x2,y1)
u)
(0,0)
(x1,y2)
(u1,0)
(x2,y2)

Figura 6.14

No es menester explicar detalladamente ambos sistemas. Remitimos a la bibliografía especializada que puede encontrar en los apéndices, o al resumen que de ellos expusimos en el Capítulo II.

Lo que haremos a continuación es explicar cómo hemos hecho para la visualización de las variables. Una cosa que tienen en común ambos sistemas gráficos es que suponen la ventana (para X-Windows) o la pantalla (para BGI) dividida en (xl+1) por (yl + 1) pixels (ver figura 6.15), localizados de izquierda a derecha (eje **x**) y de arriba a abajo (eje **y**). Así, un pixel quedará determinado por el par de coordenadas (x,y). xl e yl son los últimos pixels de cada eje: para BGI, corresponden a las dimensiones en pixels de la tarjeta gráfica, y para X-Windows, las dimensiones de la ventana.

Si nuestra curva es una representación de la variable **v** frente a la variable **u**, basta con hacer los siguientes cambios de coordenadas, para su representación mediante las funciones que dibujan puntos (o líneas), y que trabajan con (x, y), putpixel(x, y, color) en BGI y XDrawPoint(. . . , x, y) en X Windows :

$$x = \frac{x_2 - x_1}{u_1} u + x_1 \qquad y = y_2 - \frac{y_2 - y_1}{v_1} v$$

donde u1 y v1 son los límites de las magnitudes u y v respectivamente.

VI.4.- PROGRAMA PLANTA.EXE.-

VI.4.1.- CONSIDERACIONES PREVIAS.-

Este programa es la versión para PC de nuestro simulador de planta de vapor. Debido a las limitaciones en velocidad de proceso de un 80386 (en comparación con estaciones de trabajo), y necesitando una gran velocidad de cálculo para nuestra simulación, el resultado es claramente inferior a la misma aplicación realizada sobre una estación de trabajo (aplicación *planta*). Por poner un ejemplo, realizando la misma simulación (el mismo sistema 3, los mismos valores iniciales, y con tf=1000. s, o sea, 16' 40") mediante PLANTA.EXE y *planta* sobre distintas máquinas, se obtuvieron los siguientes tiempos totales de simulación:

Programa :	PLANTA.EXE	PLANTA.EXE	PLANTA.EXE	*planta*
Máquina :	80386 25 MHz sin coprocesador	80386 33 MHz con coprocesador	80486 33 MHz	Sun Sparc 1+
Tiempo :	45 ' 3 "	9 '	4 ' 15 "	37 "

Por otro lado, el entorno gráfico empleado (BGI) es de una potencia y de un impacto visual inferior al X-Windows utilizado para *planta*, el cual goza además de una facilidad de interacción con el teclado y el ratón mucho mayor.

Por todo ello, consideramos a PLANTA.EXE como una primera aproximación al problema de la simulación de una planta de vapor. Era necesario hacerlo por dos motivos: en primer lugar, porque la amplia difusión del PC nos obligaba a disponer de una versión que actuase sobre estas máquinas para el uso por parte de los estudiantes

o profesionales interesados, y por otro, como "test" de pruebas para la construcción de una aplicación más completa y potente, sobre máquinas superiores al PC.

Debido a esta inferioridad de proceso de PLANTA.EXE frente a planta, y debido también a su provisionalidad como fuente de pruebas, no hemos querido dotarla de vistosos menús y gran interactividad, ofreciendo tanto una presentación espartana (menús de pantallas de texto) como poca variedad gráfica (pocas ventanas, etc.). También por ello no vamos a extendernos en la descripción del código, pues la mayor parte de éste será usado por *planta*; remitimos pues al estudio de *planta* como una versión más acabada y perfecta, limitándonos a continuación a ofrecer las características (herramientas tecnológicas usadas y códigos desarrollados) de la aplicación, y una guía de uso para su manejo. Para no duplicar contenidos, no ofreceremos resultados de simulaciones, dejando los ejemplos de éstas para el apartado VI.6, donde se analizarán los obtenidos por la aplicación *planta*.

VI.4.2.- CARACTERISTICAS.-

- **HERRAMIENTAS UTILIZADAS.**

  En la tabla 4.1 encontramos un resumen de algunas características de las dos aplicaciones. Podemos ver cómo, para PLANTA.EXE hemos utilizado los siguientes elementos :

  — PC 80386 (ver apartado II.3.1), SVGA color.

  — Compilador Turbo C++ v1.0 (ver apartado II.3.2), con entorno gráfico Borland Graphic Interface (BGI).

- **DESCRIPCION DEL CODIGO.**

  A continuación ofrecemos la composición del código de esta aplicación, describiendo los ficheros considerados, sus tamaños y contenidos fundamentales. En total, tenemos un fichero cabecera (*.H) y 9 fuentes (*.CPP). El ejecutable es PLANTA.EXE, el cual se genera linkando los ficheros objeto. La generación de estos últimos, y su linkado, se efectúa mediante el fichero PLANTA.PRJ, creado por el propio entorno del compilador.

FICHERO	TIPO	TAMAÑO (Kb)	DESCRIPCION
CENTRAL.EXE	ejecutable	202.3	Fichero ejecutable.
CENTRAL.H	cabecera	13.9	Definición de clases y funciones.
CENTRAL.CPP	fuente	7.1	Simulación de sistemas 1 y 2.
CENTRALP.CPP	fuente	4.7	Simulación de sistema 3.
CLASES1.CPP	fuente	5.4	Definición de métodos de clases.
CLASES2.CPP	fuente	13.7	Definición de métodos de clases.
CLASES3.CPP	fuente	3.2	Definición de métodos de clases.
FUNCIONS.CPP	fuente	11.8	Funciones de gestión de tablas.
GRAFICOS.CPP	fuente	18.3	Funciones para representaciones gráficas.
TABLAS.CPP	fuente	15.2	Tablas de vapor (1ª parte).
TABLAS1.CPP	fuente	17.4	Tablas de vapor (2ª parte).

VI.4.3.- GUIA DE USO.-

El programa se arranca desde el sistema operativo DOS mediante el mandato :

A:\> PLANTA

donde A, B o C es la unidad lógica donde se encuentra el programa PLANTA.EXE .

En la figura 6.15 podemos ver la estructura de pantallas que se suceden a modo de menús. Son sencillas pantallas en modo texto. En primer lugar se nos solicita el sistema que queremos simular. Elegido éste, se nos ofrece una nueva pantalla, en la que se nos pide si vamos a generar la simulación o vamos a recuperarla a partir de un fichero de datos.

Para el caso de mostrar un fichero de simulación, hemos de especificar su identificador, aportando tan sólo el número **y**, mediante la sintaxis :

simxy.dat

donde **x** es el número del sistema (1, 2 o 3), e **y** identifica la simulación.

En el caso de haber especificado que queremos generar una simulación, aparecerá una pantalla con una serie de parámetros que muestran algunos valores iniciales de la simulación (otros están definidos en el código y no podemos cambiarlos mas que manipulando el código y volver a compilarlo). Estos parámetros cambian según el sistema considerado (ya que algunas variables pertenecen a dispositivos que no existen en algún sistema). De todas formas, los parámetros fundamentales (como son los que establecen los tiempos) son fijos. Tenemos la posibilidad de cambiar algunos de estos valores, dando el índice que identifica a la variable, y su nuevo valor. Cuando ya tengamos los valores iniciales requeridos, le decimos al programa que deseamos continuar.

Entonces se nos muestra la opción de si deseamos guardar la simulación, que vamos a generar, en un fichero. En caso afirmativo, hemos de fijar cada cuántos pasos de simulación vamos a coger datos y escribirlos sobre el fichero (hit), mostrando a continuación los valores de las iteraciones totales y el número de grupos de datos que se guardarán (ver VI.3.2). Terminada esta operación, podemos continuar o volver al anterior menú de parámetros iniciales; en este último caso, se destruiría el fichero creado. En caso de continuar, llegaremos al menú de ventanas.

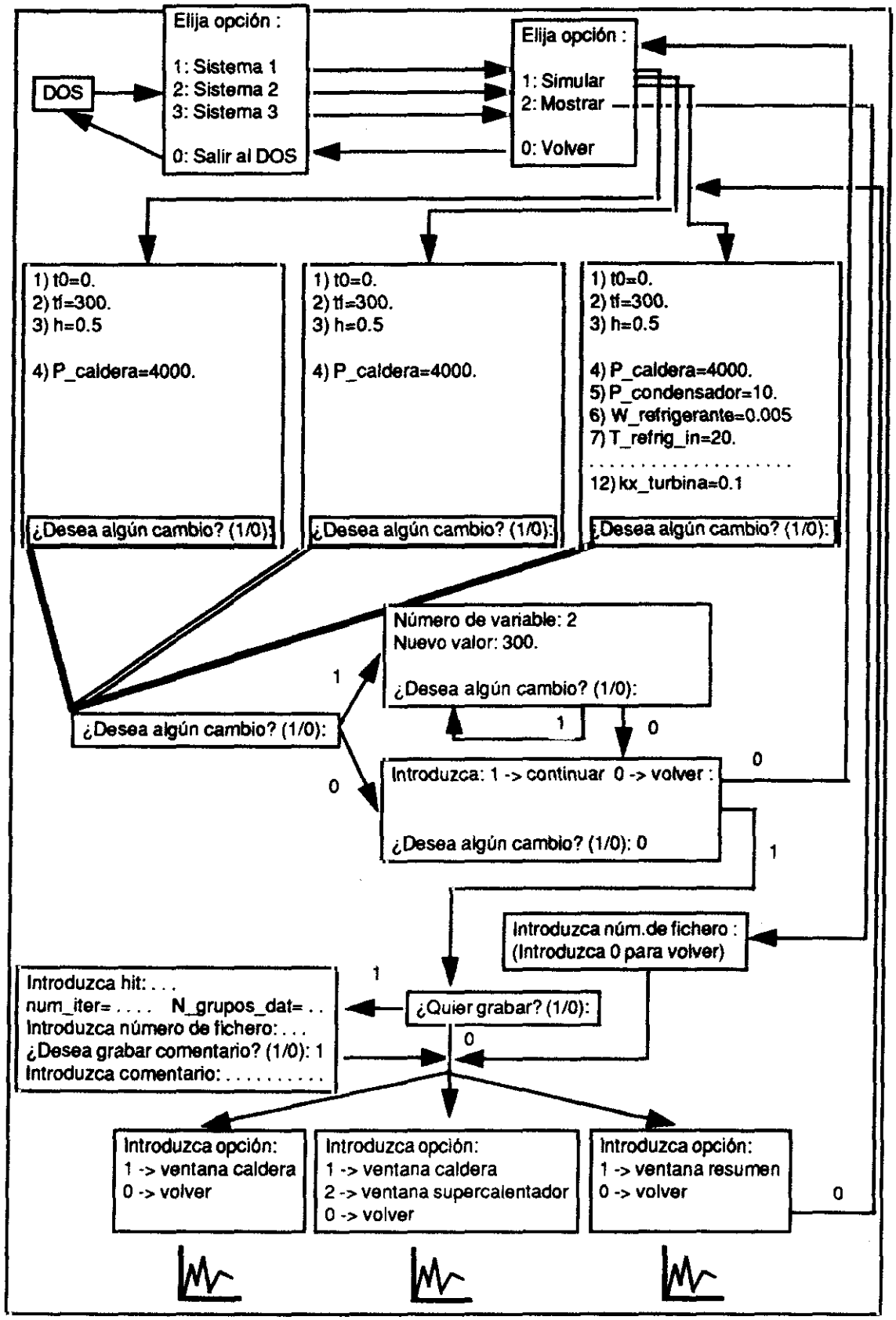

Elija opción :
1: Sistema 1
2: Sistema 2
3: Sistema 3
0: Salir al DOS
DOS
Elija opción :
1: Simular
2: Mostrar
0: Volver
1) t0=0.
2) tf=300.
3) h=0.5
4) P_caldera=4000.
¿Desea algún cambio? (1/0):
1) t0=0.
2) tf=300.
3) h=0.5
4) P_caldera=4000.
¿Desea algún cambio? (1/0):
1) t0=0.
2) tf=300.
3) h=0.5
4) P_caldera=4000.
5) P_condensador=10.
6) W_refrigerante=0.005
7) T_refrig_in=20.
12) kx_turbina=0.1
¿Desea algún cambio? (1/0):
Número de variable: 2
Nuevo valor: 300.
¿Desea algún cambio? (1/0):
¿Desea algún cambio? (1/0):
Introduzca: 1 -> continuar 0 -> volver :
¿Desea algún cambio? (1/0): 0
Introduzca núm.de fichero :
(Introduzca 0 para volver)
Introduzca hit: . . .
num_iter= . . . . N_grupos_dat= . .
Introduzca número de fichero: . . .
¿Desea grabar comentario? (1/0): 1
Introduzca comentario: . . . . . .
¿Quier grabar? (1/0):
Introduzca opción:
1 -> ventana caldera
0 -> volver
Introduzca opción:
1 -> ventana caldera
2 -> ventana supercalentador
0 -> volver
Introduzca opción:
1 -> ventana resumen
0 -> volver

Figura 6.15

Al menú de ventanas se puede llegar, por tanto, por tres caminos:

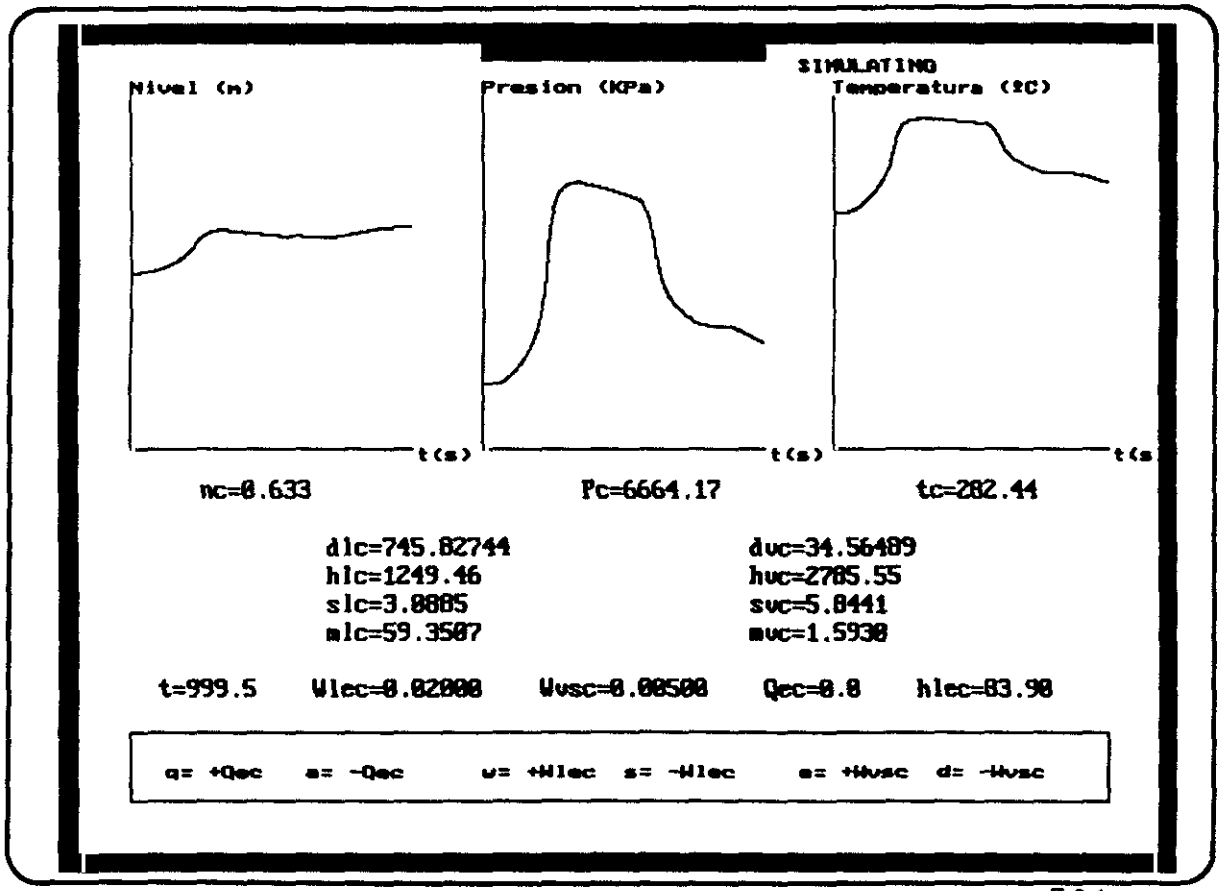
— Generando una simulación sin grabar datos en fichero.

— Generando una simulación guardando datos en fichero.

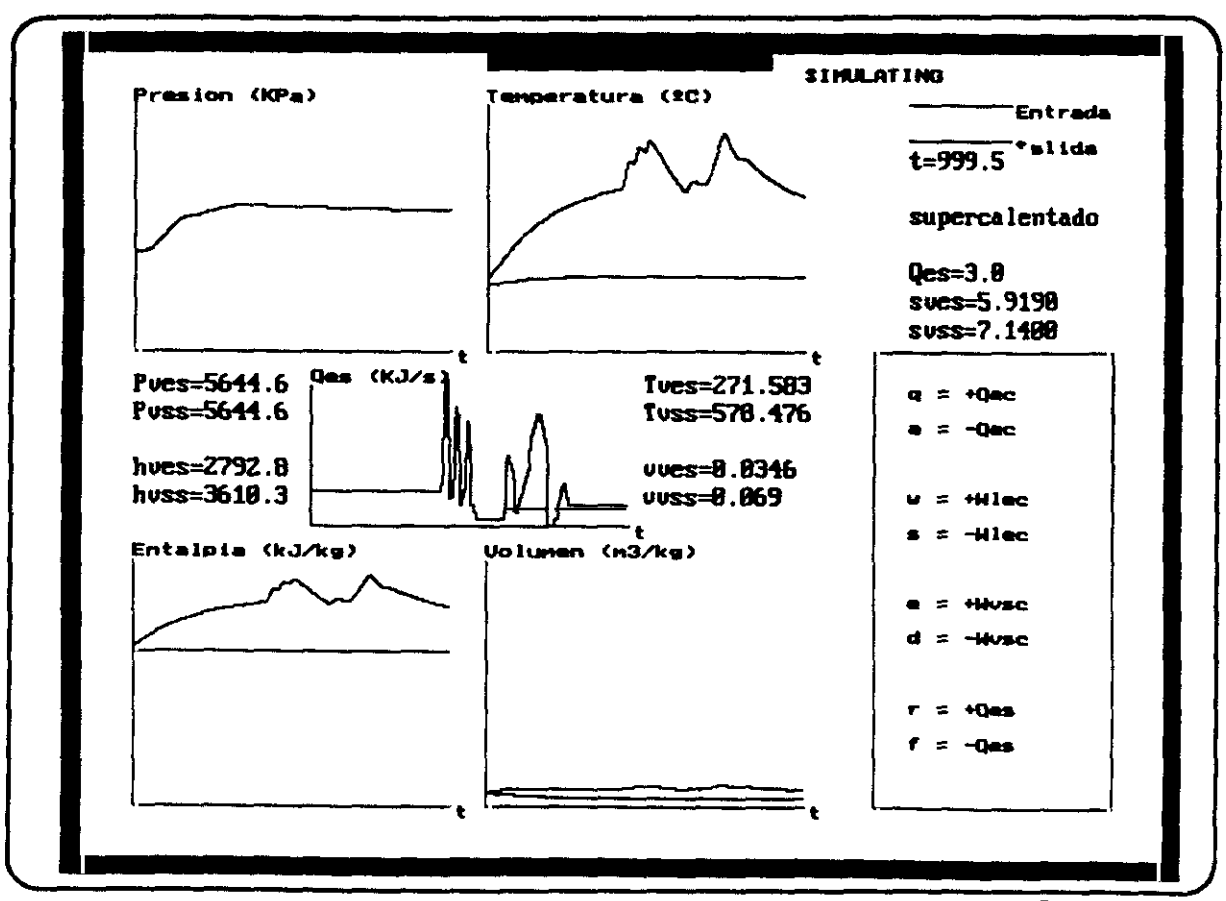
— Mostrando la simulación guardada en un fichero.

En este menú vamos a especificar en qué tipo de ventana (pantalla) queremos visualizar las gráficas. Por ejemplo, para el sistema 2 hay dos posibles pantallas (ver fotografías F 6.1 y F 6.2), una mostrando gráficas de la caldera y otra mostrando las del supercalentador. En los otros sistemas no hay este tipo de opción, disponiendo tan sólo de una pantalla.

En las pantallas se monitorizan ciertas variables, pudiendose manipular las de control mediante la pulsación de determinados botones del teclado, como se puede ver en un recuadro informativo.



F 6.1



F 6.2

Figura 6.16

VI.5.- PROGRAMA PLANTA.-

VI.5.1.- CONSIDERACIONES PREVIAS.-

El programa *planta* es un simulador de los sistemas termodinámicos considerados en el apartado VI.1, de los cuales el último es nuestro prototipo de planta de vapor.

Tras la experiencia en el desarrollo de PLANTA.EXE para PCs surgió la necesidad de contar con una versión para estaciones de trabajo con X-windows; la razón de ello era que el programa para PC tenía ciertas limitaciones: velocidad de proceso lenta para las simulaciones (ver tabla 6.1), entorno gráfico (BGI) poco sofisticado, etc. Por eso decidimos no dotar a PLANTA.EXE de muchos refinamientos y dejarla como "borrador" o "banco de pruebas" para una versión definitiva, sofisticada y potente, que fué *planta*.

Así, partiendo de los conocimientos y desarrollos de PLANTA.EXE, incorporamos:

— Mejoras en la presentación de gráficos, menús, etc.

— Ventanas de ayudas y explicaciones.

— Interactividad con el ratón y el teclado sobre menús, botones, controles, etc.

— Gestión visual e interactiva de los ficheros de simulación.

VI.5.2.- CARACTERISTICAS.-

- **HERRAMIENTAS UTILIZADAS.**

En la tabla 4.1 podemos encontrar un resumen de algunas características de las aplicaciones. Para *planta*, se consideraron los siguientes elementos :

— Estación de trabajo Sun Sparc 1+ (ver apartado II.3.1).

— Compilador GNU g++ versión 1.39 (ver apartado II.3.2).

— MIT X-Windows v11 R4 (ver apartado II.3.2).

- **DESCRIPCION DEL CODIGO.**

A continuación ofrecemos la composición del código de esta aplicación, describiendo los ficheros considerados, sus tamaños y contenidos fundamentales. En total, tenemos un ejecutable, un fichero Makefile, tres ficheros cabecera (*.h) y 23 fuentes (*.c). Serán compilados por g++ mediante el fichero **Makefile**, que crea los 23 ficheros objetos (*.o) y los linka (junto con librerías X) para obtener el ejecutable **planta**.

FICHERO	TIPO	TAMAÑO (Kb)	DESCRIPCION
planta	ejecutable		Fichero ejecutable, bajo X-windows.
Makefile	proyecto		Especifica la forma de compilación.
clases.h	cabecera	10.4	Contiene declaraciones de clases (POO).
theIcon.h	cabecera	3.2	Especifica el dibujo del icono X.
windows.h	cabecera	9.2	Declaración de todas las funciones.
argsx.c	fuente	4.3	Argumentos para la ejecución de aplicaciones X.
ayudas.c	fuente	11.2	Funciones de ayuda y explicación.
clases.c	fuente	8.5	Definición de las clases.
colorx.c	fuente	3.9	Gestión de colores en X.
cursores.c	fuente	1.6	Definición de los cursores empleados.
datos.c	fuente	2.5	Definición de datos diversos.
dibujos.c	fuente	21.3	Funciones que dibujan gráficos.
drawx.c	fuente	4.5	Funciones para dibujar en X.
esquema1.c	fuente	10.6	Simulación del sistema 1.
esquema2.c	fuente	12.3	Simulación del sistema 2.

FICHERO	TIPO	TAMAÑO (Kb)	DESCRIPCION
esquema3.c	fuente	19.6	Simulación del sistema 3.
eventos.c	fuente	0.7	Funciones para gestionar los eventos.
funcions.c	fuente	11.5	Funciones de gestión de las tablas de vapor.
initx.c	fuente	2.7	Inicialización del entorno X.
main.c	fuente	3.9	Estructura general del programa (menús).
menus.c	fuente	15.5	Funciones para la gestión de los menús.
mostrar.c	fuente	11.2	Simulación mediante ficheros de datos.
quitx.c	fuente	0.9	Salida del entorno X.
raton.c	fuente	6.9	Gestión del ratón.
tablas.c	fuente	32.1	Tablas de vapor (definición de los arrays).
teclado.c	fuente	7.3	Gestión del teclado.
textx.c	fuente	1.3	Manejo de textos en X.
windowx.c	fuente	5.8	Creación y propiedades de las ventanas en X.

Con el objeto de facilitar la descripción del código y orientar al lector interesado en su estudio, ofrecemos a continuación la lista de clases, funciones y otros elementos de interés definidos (no declarados) en cada fichero, citando nombre, tipo y descripción:

— **Nombre:**
es el identificador.

— **Tipo:**
si es clase, función, variable global, array, macro, estructura, etc. No daremos aquí ninguna información sobre valores que devuelven las funciones, argumentos o variables locales de las mismas, contenidos de las clases (variables, métodos, etc.), tipo y longitud de los arrays, etc.; todo ello se encuentra en los propios listados que se ofrecen en los apéndices.

— **Descripción:**
es un mero comentario orientativo acerca del significado de cada elemento.

clases.h :

Nombre	Tipo	Descripción
Liquido_saturado	clase	Implementa el estado de líquido saturado
Vapor_saturado	clase	Id. estado de vapor saturado
Vapor	clase	Id. estado de vapor.
Valvula	clase	Representa la válvula.
Fuente_de_liquido	clase	Id. la fuente de líquido
Quemador	clase	Id. un quemador.
Caldera	clase	Id. una caldera.
Supercalentador	clase	Id. un supercalentador.
Turbina	clase	Id. una turbina.
Bomba	clase	Id. una bomba
Condensador	clase	Id. un condensador

ayudas.c :

Nombre	Tipo	Descripción
ayuda	Función	Gestiona las ventanas de ayuda general y ayudas de sistemas.
expose_ayuda	Función	Dibuja sobre la ventana de ayuda general.
expose_ayuda1	Función	Dibuja sobre la ventana de ayuda del sistema 1.
expose_ayuda2	Función	Dibuja sobre la ventana de ayuda del sistema 2.
expose_ayuda3	Función	Dibuja sobre la ventana de ayuda del sistema 3.
ayuda_modo	Función	Gestiona la ventana de ayuda del menú de modo.
ayuda_grabación	Función	Gestiona la ventana de ayuda del menú de grabación.
ayuda_mostrar	Función	Gestiona la ventana de ayuda del menú de .mostrar.

clases.c :

Nombre	Tipo	Descripción
xxxxxxx	Métodos públicos	Definición de varios métodos públicos de las clases definidas en el fichero clases.h

colorx.c :

Nombre	Tipo	Descripción
nombres_colores	Array	Describe los nombres de 66 colores (0 -> 65)
set_color_nombre	Función	Selecciona el color (foreground) dando su nombre.
set_color	Función	Selecciona el color dando su número de identificación.
init_colores_defecto	Función	Selecciona los colores por defecto del Hardware.

cursores.c :

Nombre	Tipo	Descripción
???????	Estructura Cursor	Cursores que empleamos.
cursores	Función	Crea y define los cursores
limpiar_cursores	Función	Limpia X de cursores.

datos.c :

Nombre	Tipo	Descripción
??????	Estructura XPoint	Diversas declaraciones y definiciones de conjuntos de puntos que serán usados en numerosos dibujos.

dibujos.c :

Nombre	Tipo	Descripción
boton1	Función	Dibuja un tipo de botón (rectangular).
boton2	Función	Dibuja un tipo de botón (circular):
boton3	Función	Dibuja un tipo de botón (flecha izquierda).
boton4	Función	Dibuja un tipo de botón (flecha derecha).
decorado_menu_inicial	Función	Dibuja el cuadro con el paisaje de la central.
expose_menu_inicial	Función	Dibuja el menú inicial (decorado, botones, etc.).
expose_menu_parametros	Función	Dibuja el menú de parámetros.
expose_menu_mostrar	Función	Dibuja el menú de mostrar simulaciones de ficheros.
imprimo_parametros	Método	Método de la clase Scheduler para imprimir los parametros.
expose_menu_grabacion	Función	Dibuja el menú de grabación de simulaciones en ficheros.
expose_control	Función	Dibuja el menú de control de la simulación.
dibujo_entorno	Función	Dibuja la ventana de monitorización de variables.
dibujo_punto	Función	Dibuja un punto de color en cierta lugar de la ventana.
pintar_caldera	Función	Dibuja una caldera.
pintar_quemador	Función	Dibuja un quemador.
pintar_valvula	Función	Dibuja una válvula.
pintar_supercalentador	Función	Dibuja u supercalentador.
pintar_turbina	Función	Dibuja una turbina.
pintar_condensador	Función	Dibuja un condensador.
pintar_bomba	Función	Dibuja una bomba.
pintar_fuente	Función	Dibuja una fuente.
pintar_flecha_arr	Función	Dibuja una flecha hacia arriba.
pintar_flecha_abj	Función	Dibuja una flecha hacia abajo.
pintar_flecha_izq	Función	Dibuja una flecha hacia la izquierda.
pintar_flecha_der	Función	Dibuja una flecha hacia la derecha.

drawx.c :

Nombre	Tipo	Descripción
dibujaLinea	Función	Dibuja una línea.
dibujaPunto	Función	Dibuja un punto.
dibujaRectangulo	Función	Dibuja un rectángulo vacío.
llenaRectangulo	Función	Dibuja un rectángulo lleno.
dibujaOvalo	Función	Dibuja una elipse vacía.
llenaOvalo	Función	Dibuja una elipse llena.
dibujaArco	Función	Dibuja un arco.
llenaArco	Función	Dibuja un arco, llenando el ángulo que forma.
dibujaPuntos	Función	Dibuja un conjunto de puntos.
dibujaLineas	Función	Dibuja un conjunto de líneas unidas por los extremos.
dibujaArcos	Función	Dibuja un conjunto de arcos.
llenaArcos	Función	Dibuja un conjunto de arcos llenos.
dibujaRectangulos	Función	dibuja un conjunto de rectángulos vacíos.
llenaRectangulos	Función	Dibuja un conjunto de rectángulos llenos.
dibujaSegmentos	Función	Dibuja un conjunto de líneas.
llenaPoligono	Función	Dibuja un polígono lleno.
CIRCULO_TOTAL	Macro	Especifica el ángulo para el círculo (360º).
EMPIEZA_ANGULO	Macro	Especifica la posición desde la que se cuenta el ángulo.

esquema1.c :

Nombre	Tipo	Descripción
esquema_1	Método.	Scheduler: Gestiona la simulación del sistema 1.
abrir_controles_esquema1	Función	Abre y dibuja las ventanas de los botones de control.
cierro_controles	Función	Cierra las ventanas de los botones de control.
imprimo_controles_esquema1	Función	Imprime el valor de las variables de control en cada t.
abrir_ventanas_esquema1	Función	Abre y dibuja las ventanas de monitorización.
cierro_ventanas_esquema1	Función	Cierra las ventanas de monitorización.

esquema2.c :

Nombre	Tipo	Descripción
esquema_2	Método.	Scheduler: Gestiona la simulación del sistema 2.
abrir_controles_esquema2	Función	Abre y dibuja las ventanas de los botones de control.
imprimo_controles_esquema2	Función	Imprime el valor de las variables de control en cada t.
abrir_ventanas_esquema2	Función	Abre y dibuja las ventanas de monitorización.
cierro_ventanas_esquema2	Función	Cierra las ventanas de monitorización.

esquema3.c :

Nombre	Tipo	Descripción
esquema_3	Método.	Scheduler: Gestiona la simulación del sistema 3.
abrir_controles_esquema3	Función	Abre y dibuja las ventanas de los botones de control.
imprimo_controles_esquema3	Función	Imprime el valor de las variables de control en cada t.
abrir_ventanas_esquema3	Función	Abre y dibuja las ventanas de monitorización.
cierro_ventanas_esquema3	Función	Cierra las ventanas de monitorización.

eventos.c :

Nombre	Tipo	Descripción
EV_MASK	Macro	Máscara de eventos.
inicia_eventos	Función	Prepara la ventana para los eventos definidos en la máscara.

funcions.c :

Nombre	Tipo	Descripción
intrplcn	Función	Halla y[i]=f(x[i]) dados x[i-1], x[i+1], y[i-1] e y[i+1]
tabla_1d	Función	Halla y0=f(x0) dada una curva tabulada.
tabla_2d_xy	Función	Halla z0 conocidos x0 e y0 en una tabla x * y
tabla_2d_xz	Función	Halla y0 conocidos x0 y z0 en una tabla x * z
tabla_2d_xy_sh	Función	Halla z0 conocidos x0 e y0 en una tabla x*y (supercalentado)
tabla_2d_xz_sh	Función	Halla y0 conocidos x0 y z0 en una tabla x*z (supercalentado).
tabla_2d_xy_liq	Función	Halla z0 conocidos x0 e y0 en una tabla x * y (líquido).
tabla_2d_xz_liq	Función	Halla y0 conocidos x0 y z0 en una tabla x * z (líquido).

initx.c :

Nombre	Tipo	Descripción
el_display	Estructura Display	Display para la aplicación.
el_screen	Variable	Pantalla para la aplicación.
profundidad	Variable	Número de planos de color.
pixel_blanco, pixel_negro	Variables	Colores blanco y negro del sistema.
mapa_color	Estructura Colormap	Mapa de color por defecto del sistema.
el_cursor	Estructura Cursor	Cursor del programa de aplicación.
ventana_raiz	Estructura Window	Ventana padre o raíz del programa.
incia_X	Función	Establece la conexión con el servidor X.
info_X	Función	Obtiene e imprime información sobre X.

main.c :

Nombre	Tipo	Descripción
el_GC	Estructura GC	Contexto gráfico del programa.
w, w1, wp, ...	Estructura Window	Todas las ventanas usadas en nuestra aplicación.
main	Función	Función principal del programa.
start	Método	Método del Scheduler que gestiona la estructura de menús.

menus.c :

Nombre	Tipo	Descripción
abro_menu_inicial	Método	Scheduler: Abre y dibuja as ventanas del menú inicial.
menu_inicial	Método	Scheduler: Gestiona el menú inicial.
abro_menu_modo	Método	Scheduler: Abre y dibuja las ventanas del menú de modo.
menu_modo	Método	Scheduler: Gestiona el menú de modo.
cierro_menu_modo	Método	Scheduler: Cierra las ventanas del menú de modo.
abro_menu_parametros	Método	Scheduler: Abre, dibuja e inicializa las ventanas de este menú.
menu_parametros	Método	Scheduler: Gestiona el menú de parámetros.
cierro_menu_parametro	Método	Scheduler: Cierra las ventanas del menú de parámetros.
abro_menu_mostrar	Método	Scheduler: Abre, dibuja e inicializa este menú.
menu_mostrar	Método	Scheduler: Gestiona el menú de mostrar simulaciones.
cierro_menu_mostrar	Método	Scheduler: Cierra las ventanas de este menú de ficheros.
set_fichero_mostrar	Método	Scheduler: Selecciona el fichero de simulación a mostrar.
abro_menu_grabacion	Método	Scheduler: Abre y dibuja el menú de grabación en ficheros.
menu_grabacion	Método	Scheduler: Gestiona el menú de grabación de ficheros.
cierro_menu_grabacion	Método	Scheduler: Cierra las ventanas dil menú de grabación.
set_grabacion	Método	Scheduler: Gestiona el menú de grabación de ficheros.
creo_fichero	Método	Scheduler: Crea el fichero de datos.
abro_control	Método	Scheduler: Abre y dibuja el menú de control de la simulación.
cierro_control	Método	Scheduler: Cierra el menú de control de la simulación.
control	Método	Scheduler: Gestiona el menú de control de la simulación.

mostrar.c :

Nombre	Tipo	Descripción
muestra_simulación	Método	Scheduler: Gestiona la simulación guardada en un fichero.
esquema_mostrar	Método	Scheduler: Lee datos del fichero y representa.

quitx.c :

Nombre	Tipo	Descripción
quitar_X	Función	Cierra la conexión con el servidor X.
limpia_ventanas_GCs	Función	Cierra las ventanas y los contextos gráficos de X.

raton.c :

Nombre	Tipo	Descripción
raton_menu_inicial	Método	Scheduler: Gestiona el ratón para el menú inicial.
raton_menu_modol	Método	Scheduler: Gestiona el ratón para el menú de modo.
raton_menu_parametros	Método	Scheduler: Gestiona el ratón para el menú de parametros.
raton_menu_mostrar	Método	Scheduler: Gestiona el ratón para el menú de mostrar.
raton_menu_grabacion	Método	Scheduler: Gestiona el ratón para el menú de grabación.
raton_set_grabacion	Método	Scheduler: Gestiona el ratón para dar hit y el núm. de fichero.
raton_control	Método	Scheduler: Gestiona el ratón para el menú de control.
teclado_control	Método	Scheduler: Gestiona el teclado para el menú de control.
raton_seguir	Método	Scheduler: Gestiona el ratón para continuar la simulación.
raton_ayuda_modo	Función	Gestiona el ratón para la ventana de ayuda del menú de modo.
raton_ayuda_grabacion	Función	Gestiona el ratón para la ayuda del menú de grabación.
raton_ayuda_mostrar	Función	Gestiona el ratón para la ayuda del menú de mostrar.
raton_ayuda	Función	Gestiona el ratón para la ventana de ayuda general.

tablas.c :

Nombre	Tipo	Descripción
orden_interp_sat	Constante	Describe un orden de interpolación para el vapor saturado.
a_sat	Array	Datos par interpolar curvas.
c_sat	Array	Datos para interpolar curvas.
T_sat	Array	Temperaturas del vapor líquido saturado.
P_sat	Array	Presiones del vapor y líquido saturado.
h_sat_v	Array	Entalpías del vapor saturado.
s_sat_v	Array	Entropías del vapor saturado.
v_sat_v	Array	Volúmenes específicos del vapor saturado.
P_l	Array	Presiones del líquido.
T_l	Array	Temperaturas del líquido.
h_l	Array	Entalpías del llíquido.
v_l	Array	Volúmenes específicos del líquido.
orden_interp_liq	Constante	Orden de interpolación para datos del líquido.
e_liq	Array	Datos del líquido para interpolar.
T_dat	Array	Temperaturas considaradas en las tablas del líquido.
b_sat	Array	Datos del líquido saturado para interpolar.
d_sat	Array	Datos del líquido saturado para interpolar.
s_sat_l	Array	Entropías del líquido saturado.
v_sat_l	Array	Volúmenes específicos del líquido saturado.
h_sat_l	Array	Entalpías del líquido saturado.
P_sup	Array	Presiones del vapor supercalentado.
TK_sup	Array	Temperaturas del vapor supercalentado.
h_sup	Array	Entalpías del vapor supercalentado.
s_sup	Array	Entropías del vapor supercalentado.
v_sup	Array	Volúmenes específicos del vapor supercalentado.

Nombre	Tipo	Descripción
orden_interp_vap	Constante.	Orden de interpolación para curvas de vapor.
x_dat	Array	Calidades del vapor.
c_vap	Array	Datos de curvas de vapor interpoladas.
a_vap	Array	Datos de curvas de vapor interpoladas.

teclado.c :

Nombre	Tipo	Descripción
length	Variable	Longitud de una cadena.
buffer_tecla	Array	Guarda la cadena introducida por el teclado.
simbolo_tecla	Estructura KeySym	Describe el tipo de teclado.
long_max_buffer	Constante	Longitud máxima de la cadena.
procesa_tecla	Función	Procesa la cadena entrada.
teclado_menu_inicial	Función	Gestiona el teclado para el menú inicial.
teclado_menu_modo	Función	Gestiona el teclado para el menú de modo.
teclado_menu_parametros	Función	Gestiona el teclado para el menú de parámetros.
teclado_menu_mostrar	Función	Gestiona el teclado para el menú de mostrar.
teclado_menu_grabacion	Función	Gestiona el teclado para el menú de grabación.
teclado_botonOK	Función	Gestiona el teclado para el botón de OK.

textx.c :

Nombre	Tipo	Descripción
inicia_font	Array	Inicializa el conjunto de fonts.
drawImageString	Función	Dibuja una cadena de texto en formato inverso.
drawString	Función	Dibuja una cadena de texto.

windowx.c :

Nombre	Tipo	Descripción
ANCHO_BORDE	Macro	Anchura del borde de las ventanas.
crea_GC	Función	Crea un contexto gráfico.
abroVentana	Función	Selecciona parámetros y abre una ventana.

VI.5.3.- SUGERENCIAS.-

Este programa está preparado para, mediante la manipulación del código (añadiendo o cambiando elementos en los ficheros, incorporar nuevos ficheros en Makefile, etc.) y su posterior compilación, añadir una serie de características que lo harán más versátil y sofisticado. Así pues, está abierto a posibles refinamientos, para diversos estudios de simulación de plantas térmicas. Enumeramos a continuación una serie de sugerencias que se nos han ocurrido a medida que aumentaba nuestra experiencia con el programa.

- Podemos construir otros sistemas termodinámicos, mediante la conexión de dispositivos que ya están implementados en clases. Habría que escribir un fichero parecido a esquema1.c, esquema2.c y esquema3.c, teniendo cuidado de escribir las definiciones de ventanas, eventos de ratón y teclado, etc. en los ficheros correspondientes. La tarea más importante, en este caso, es la de construir el bucle de simulación que implemente de forma correcta la simulación dinámica del modelo teórico.

- También podemos quitar o añadir más ventanas de monitorización de variables, más visualización (para su manipulación) de parámetros iniciales en el menú de parámetros, etc., teniendo cuidado de especificar cada parámetro en su fichero correspondiente; para ello no hay más que seguir el ejemplo de los parámetros de inicialización existentes.

- Se puede añadir más control automático, mediante la sencilla adición de condiciones lógicas al final del bucle de simulación. Los elementos de control automático van surgiendo conforme la experiencia nos dicta cuál es el alcance de algunas variables ante ciertos comportamientos del sistema. Así por ejemplo, si vemos que el nivel de líquido en la caldera va a llegar al máximo posible, y si sabemos que haciendo Wlec -> 0 podemos evitarlo, introduciríamos algo como:

```
if ( cal.dar_nc() >= 0.95 )        bomb.set_Wb( 0. );
```

Este control automático se hace especialmente necesario cuando, bien sea porque algunas variables puedan localizarse fuera de los rangos definidos por las tablas de vapor, o bien por cualquier otra causa, se nos disparan los cálculos, llegándose a monitorizaciones anómalas.

- Podemos cambiar la cantidad por la que se incrementa o decrementa un parámetro de inicialización o uno de control, mediante la manipulación del código (en los métodos de las clases) y su posterior compilación. Las cantidades que hemos fijado son las que nos han dictado las primeras pruebas o experiencias con el simulador, lo cual no quiere decir que no sean las más adecuadas.

- Sería interesante crear un tipo de monitorización de variables que cambiase la dimensión del eje **y** cada vez que el valor de la variable representada se saliera del rango por él determinado. Si establecemos el rango del eje **y** como el máximo que pueda alcanzar la variable representada, podría ser que no apreciáramos variación en el comportamiento de la misma, mientras que si este rango fuese menor, veríamos la curva. En nuestro programa hemos considerado rangos fijos, de manera que si la variable los supera, produce un aspecto extraño; sin embargo, hemos considerado rangos suficientemente amplios como para que la variable pueda operar en ellos en un modo normal.

- Por último, pueden añadirse aspectos decorativos como una mejor presentación de los gráficos, más ayudas y explicaciones, incorporación de nuevos eventos como la pulsación continuada del ratón, etc.

VI.5.4.- GUIA DE USO.-

Nuestra aplicación se ejecuta llamando a *planta* desde el sistema operativo de nuestra estación de trabajo. Antes de describir su contenido, veamos unas nociones previas acerca de la terminología que emplearemos y del uso del ratón y el teclado:

a) Nociones previas.

Introducimos la siguiente terminología para poder entendernos en esta guía de uso del programa:

— **Botón:**
Es una ventana sobre la cual puede actuar el ratón o el teclado.

— **Menú:**
Es una ventana que contiene botones. Decimos que el menú es activo cuando es el último que se ha desplegado; en caso contrario, aunque estuviese visible, los botones de dicho menú no responden.

El ratón actúa cuando presionamos una de sus teclas permaneciendo su puntero posicionado sobre un botón (que puede tener forma rectangular, circular, etc.) considerado. El programa está preparado para que el ratón sólo pueda actuar dentro del menú activo, de forma que si presionásemos la tecla del ratón con éste posicionado sobre un botón de otro menú, entonces el botón no actuaría desplegando su ventana o efectuando su acción.

El teclado actúa sobre los botones del menú activo. Se ha de pulsar la tecla adecuada para cada botón, la cual aparece de distinto color en el texto de identificación del botón.

Una vez que hemos llamado al programa, lo primero que aparece es un:

b) Menú inicial.

Junto a un dibujo decorativo a modo de presentación, aparecen cuatro botones, representando a los tres sistemas considerados y a un menú de ayuda general (ver fotografía F 6.3). Actuando sobre cualquiera de estos botones se acciona el mismo, desplegándose el menú correspondiente. Así, podemos desde aquí entrar en cualquiera de estos menús :

— *Menú de modo:*
Si pulsamos cualquier botón de simulación de sistema.

— *Menú de ayuda general:*
Pulsando el botón de "Ayuda".

Los sistemas a simular son los tres considerados en la presente tesis; para más información, remitimos al capítulo V y al apartado VI.1.

c) Menú de ayuda general.

En él podemos ver una descripción general de las características de la simulación. Para una descripción particular de cada sistema, se nos ofrecen tres botones, mediante los cuales podemos desplegar las ayudas particulares, tal como se observa en F 6.4 .

d) Menú de modo.

Este menú nos permitirá elegir, mediante los botones correspondientes, las opciones de simular el sistema considerado generando los datos y representándolos ("simular"), o bien abrir un fichero de datos en el cual se han guardado los resultados de una simulación, para entonces representarla gráficamente ("mostrar"). Evidentemente, toda simulación guardada en un fichero de datos ha debido de ser generada (y grabada sobre el fichero) previamente mediante la opción "simular". Un tercer botón nos permitirá una breve explicación sobre este menú, y un cuarto nos ofrece la posibilidad de cerrar esta ventana para elegir otro sistema.

e) Menú de parámetros.

En la fotografía F 6.5 podemos ver deplegado el menú de parámetros sobre el de modo. Para entrar en él, pues, ha de haberse seleccionado previamente "simular" en el menú de modo. En el menú de parámetros se nos ofrece un conjunto de variables para su inicialización (esto es, para dar sus valores en el instante inicial de la simulación, t0). Vienen con unos valores prefijados, los cuales pueden variarse mediante unos botones de incremento o decremento, accionables tan sólo a través del ratón.

La cantidad mediante la cual estos parámetros incrementan o decrementan sus valores es distinta para cada variable, y ha sido determinada en función de las primeras experiencias con el simulador.

Cada sistema tiene su propio conjunto de parámetros, ya que es ilógico hablar de seleccionar el parámetro de un dispositivo que no esté presente en el sistema. La finalidad de estos parámetros es la de facilitar la experimentación del sistema, mediante sus cambios; la experiencia nos irá diciendo cuáles son más importantes (porque afectan mucho a la simulación) y cuáles no.

Para comprender el significado de cada uno de estos parámetros, en la ventana de ayuda particular del sistema correspondiente (dentro del menú de ayuda general), podrá encontrar el usuario el significado y localización de los mismos.

f) Menú de grabación.

Tras haber seleccionado los valores iniciales, al presionar "Ok" se abre el menú de grabación. Cuatro botones nos ofrecen las siguientes posibilidades:

— *"Cancelar":*
Volvemos al menú de parámetros, bien para cambiar algún valor o para salir del modo de simulación.

— *"Explicar":*
Breve descripción del propósito de este menú.

— *"No grabar":*
Presionando este botón, se abre el menú de control de simulación que inicia el proceso de la misma; "no grabar" indica que no vamos a crear un fichero donde guardar datos de la simualción.

— *"Grabar":*
Indica que deseamos grabar la simulación en un fichero. Esto conlleva tres acciones: creación y apertura de un fichero con el formato

simxy.dat,

establecimiento de los valores de grabación **x**, **y** y **hit** (al presionar "grabar", se abre un pequeño menú -ver F 6.6- donde establecemos el valor de **y** -número del fichero, que identifica a la simulación- y **hit** -ver VI.3.2-, mientras que **x** es seleccionado automáticamente al número del sistema considerado), y por último, una posterior simultaneidad de la simulación con la grabación de datos sobre el fichero. Por tanto, tras haber seleccionado los parámetros del fichero, se abre el menú de control de la simulación.

g) Menú de control de la simulación.

Tras el menú de grabación, y habiéndose seleccionado "grabar" o "no grabar", acce-demos a este menú, en el cual aparecen cuatro botones, de los cuales sólo "Empezar" y "Cancelar" son activos:

— *"Cancelar":*
Volvemos al menú de grabación. Si hubiésemos accionado este botón antes de comenzar la simulación con "Empezar", y si previamente se hubiera elegido "grabar" en el anterior menú, el fichero creado simxy.dat sería borrado.

— *"Empezar":*
Comienza la simulación con todas las características seleccionadas en los anteriores menús. Primero, se abre un **menú de variables de control**, en el cual aparece una serie de variables cuyos valores pueden ser alterados durante la simulación, mediante unos botones de incremento y decremento (y que son distintas dependiendo del sistema a simular). A continuación, se abre una serie de ventanas de monitorización de ciertas variables características del sistema (ver fotografía F 6.7); cada ventana de monitorización puede representar desde una hasta cuatro variables, cada una de ellas con un color distintivo. Tras comenzar la simulación, y antes de concluir la misma, tan sólo el botón de "Parar" es activo. Terminado el proceso de simulación, se cierra el menú de variables de control, no las ventanas de monitorización, y mediante "Cancelar" (único botón activo entonces), cerramos dichas ventanas. En este instante podemos, bien comenzar de nuevo la simulación ("Empezar"), o bien cerrar este menú y volver al anterior ("Cancelar").

— *"Parar":*
Este botón sólo es activo durante el tiempo de simulación. Efectúa una parada de la misma, de forma que podamos observar con detenimiento las gráficas o bien cancelarla mediante "Cancelar. Así, tras parar la simulación, sólo serán activos los botones de "Continuar" y "Cancelar"."

— *"Continuar":*
Este botón permite continuar con la simulación una vez que ésta ha sido parada. En cualquier otra circunstancia, este botón estará inactivo.

h) Menú de mostrar.

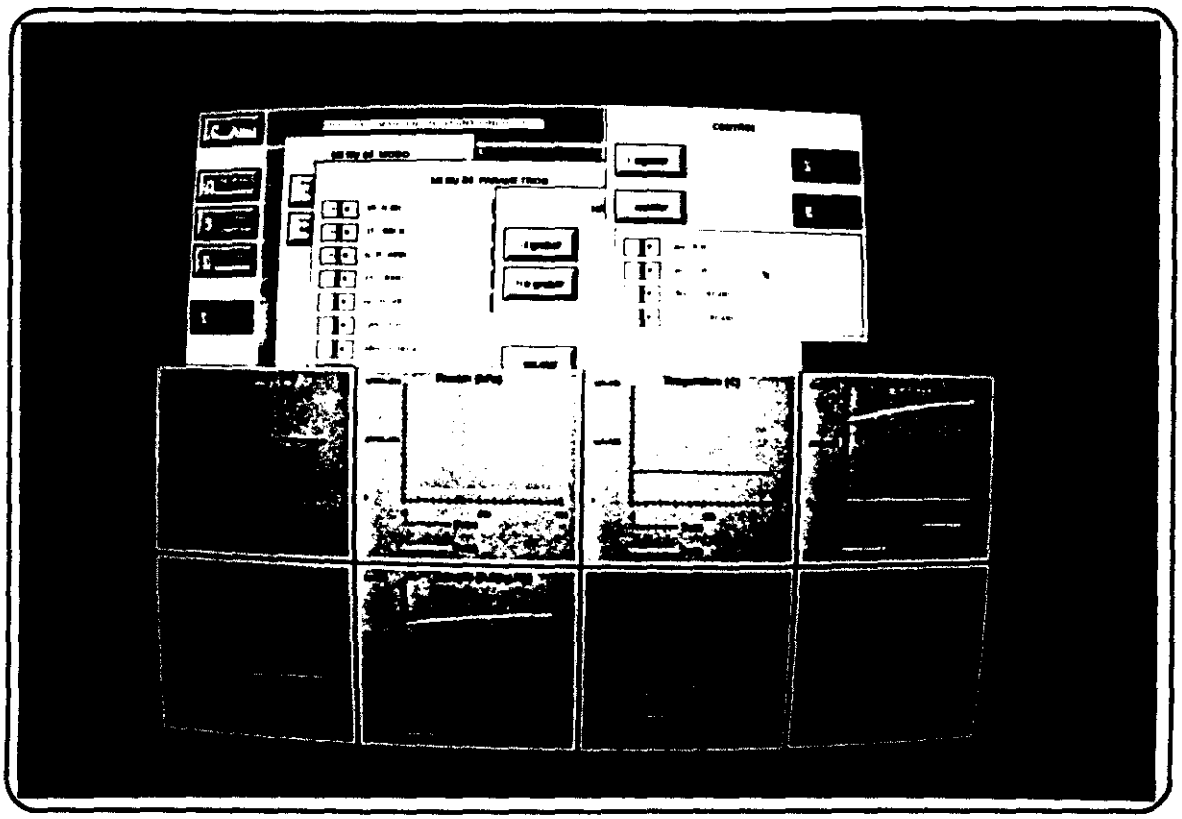
Finalmente, teníamos pendiente hablar de este menú, al cual accedemos tras haber seleccionado "Mostrar" en el menú de modo. Tenemos activos entonces cinco botones:

— *"Explicación":*
Breve descripción de la finalidad de este menú.

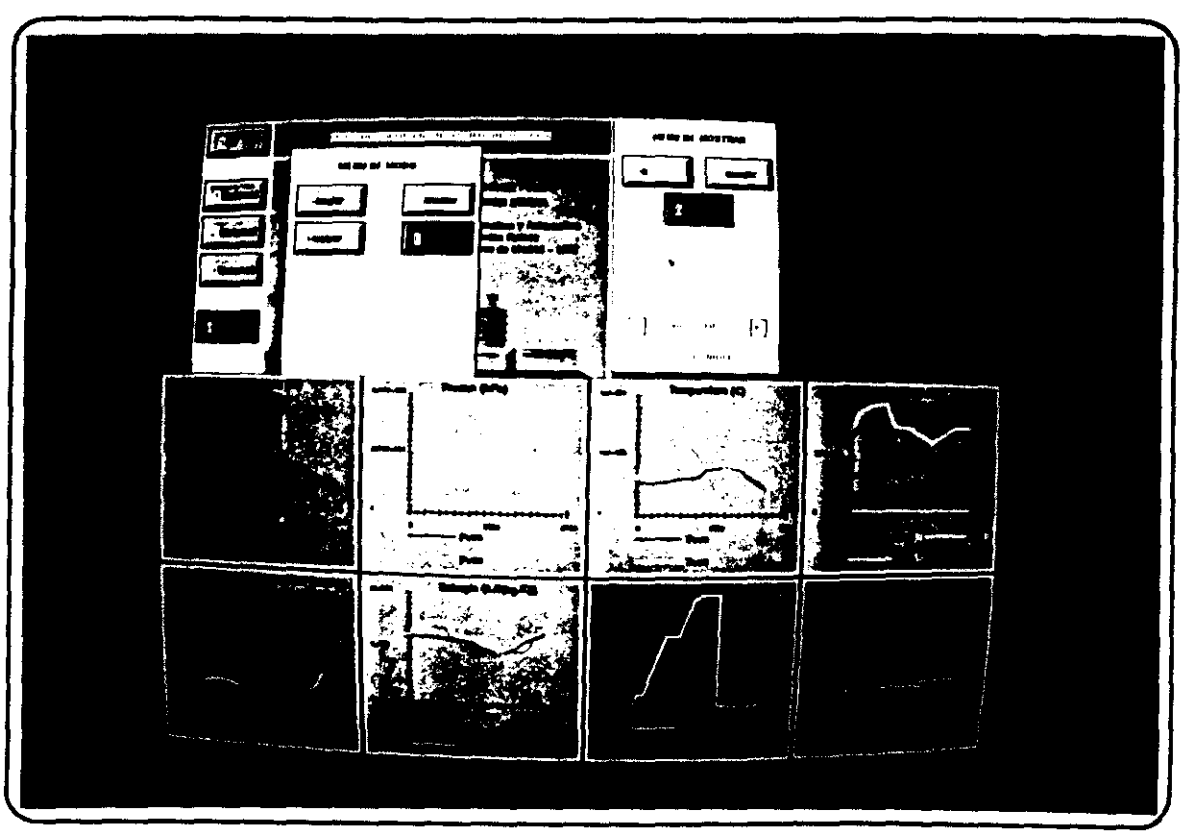
— *"Cancelar":*
Cancela el menú, retornando al menú inicial.

— *Botones "-" y +":*
Inicialmente, aparece una ventana con el fichero simx1.dat (donde x es el número del sistema considerado), bajo la cual se verá la leyenda "SI DISPONIBLE"o "NO DISPONIBLE", dependiendo de si dicho fichero existe o no en el directorio en el cual se halla la aplicación *planta*. Mediante los botones de incremento podemos ver si los ficheros simx2.dat, simx3.dat, . . ., etc. están disponibles o no.

— *"Ok":*
Comienza la monitorización de la simulación guarda en el fichero que previamente ha de haber sido seleccionado mediante "-" o "+"; una vez seleccionado éste, presionando el botón "Ok" se despliegan las ventanas de monitorización de las variables, visualizándose las curvas tal y como se generaron cuando fueron simuladas (ver F 6.8).



F 6.7



F 6.8

Figura 6.19

VI.6.- SIMULACION.-

A continuación ofreceremos algunas de las simulaciones efectuadas por la aplicación *planta*, ya que como se dijo anteriormente, es la más completa, rápida y visual de las dos aplicaciones producidas. Los siguientes ejemplos son unos cuantos entre la extensa variedad de posibilidades de experimentación de los tres sistemas considerados. A partir de estas simulaciones podremos sacar diverso tipo de información, como por ejemplo:

- ***Comportamiento ante la manipulación de controles.***
  Podremos observar cómo afectan los cambios de ciertos parámetros al funcionamiento de la planta, o cómo hemos de controlar ciertos parámetros para conseguir una respuesta requerida del sistema. Por ejemplo, si queremos aumentar el rendimiento, hemos de ver qué parámetros de control han de accionarse, y en qué cuantía.

- ***Simulación de averías.***
  Mediante la inicialización de algunas variables, o mediante el menejo de ciertos parámetros de control, podremos simular averías como, por ejemplo la pérdida excesiva de gas en la turbina a partir de Wvst=kx.Wvet,

- ***Comprensión de los fenómenos físicos producidos.***
  Observando las curvas de las variables de estado más importante, podremos comprobar de forma práctica lo que nos dice la teoría termodinámica.

- ***Otras informaciones.***
  Como de la relación que existe entre distintas variables, de qué forma y en qué cuantía se afectan los dispositivos entre sí, qué rapidez de respuesta tienen unas variables ante los eventos producidos, etc.

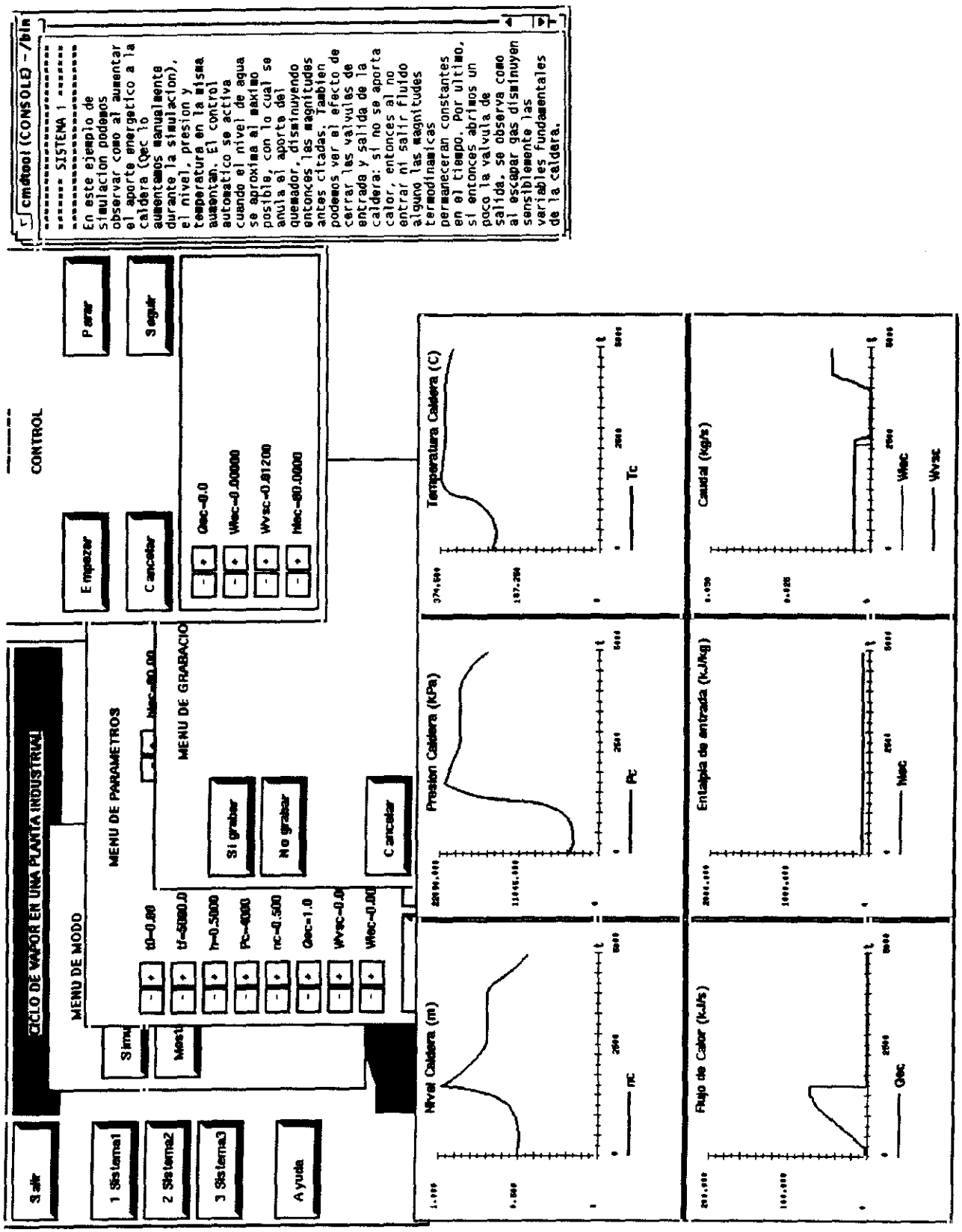

cmdtool (CONSOLE) - /bin
SISTEMA 1
En este ejemplo de simulacion podemos observar como al aumentar el aporte energetico a la caldera (Qec lo aumentamos manualmente durante la simulacion), el nivel, presion y temperatura en la misma aumentan. El control automatico se activa cuando el nivel de agua se aproxima al maximo posible, con lo cual se anula el aporte del quemador, disminuyendo entonces las magnitudes antes citadas. Tambien podemos ver el efecto de cerrar las valvulas de entrada y salida de la caldera: si no se aporta calor, entonces al no entrar ni salir fluido alguno las magnitudes termodinamicas permanecerán constantes en el tiempo. Por ultimo, si entonces abrimos un poco la valvula de salida, se observa como al escapar gas disminuyen sensiblemente las variables fundamentales de la caldera.
CONTROL
Empezar
Cancelar
Parar
Seguir
Qec=0.0
Wec=0.00000
Wvsc=0.01200
Nec=80.0000
CICLO DE VAPOR EN UNA PLANTA INDUSTRIAL
MENU DE MODO
MENU DE PARAMETROS
MENU DE GRABACIO
Si grabar
No grabar
Cancelar
Salir
1 Sistema1
2 Sistema2
3 Sistema3
Ayuda
Temperatura Caldera (C)
Tc
Caudal (kg/s)
Wec
Wvsc
Presion Caldera (kPa)
Pc
Entalpia de entrada (kJ/kg)
Hec
Nivel Caldera (m)
nc
Flujo de Calor (kJ/s)
Qec

Figura 6.20

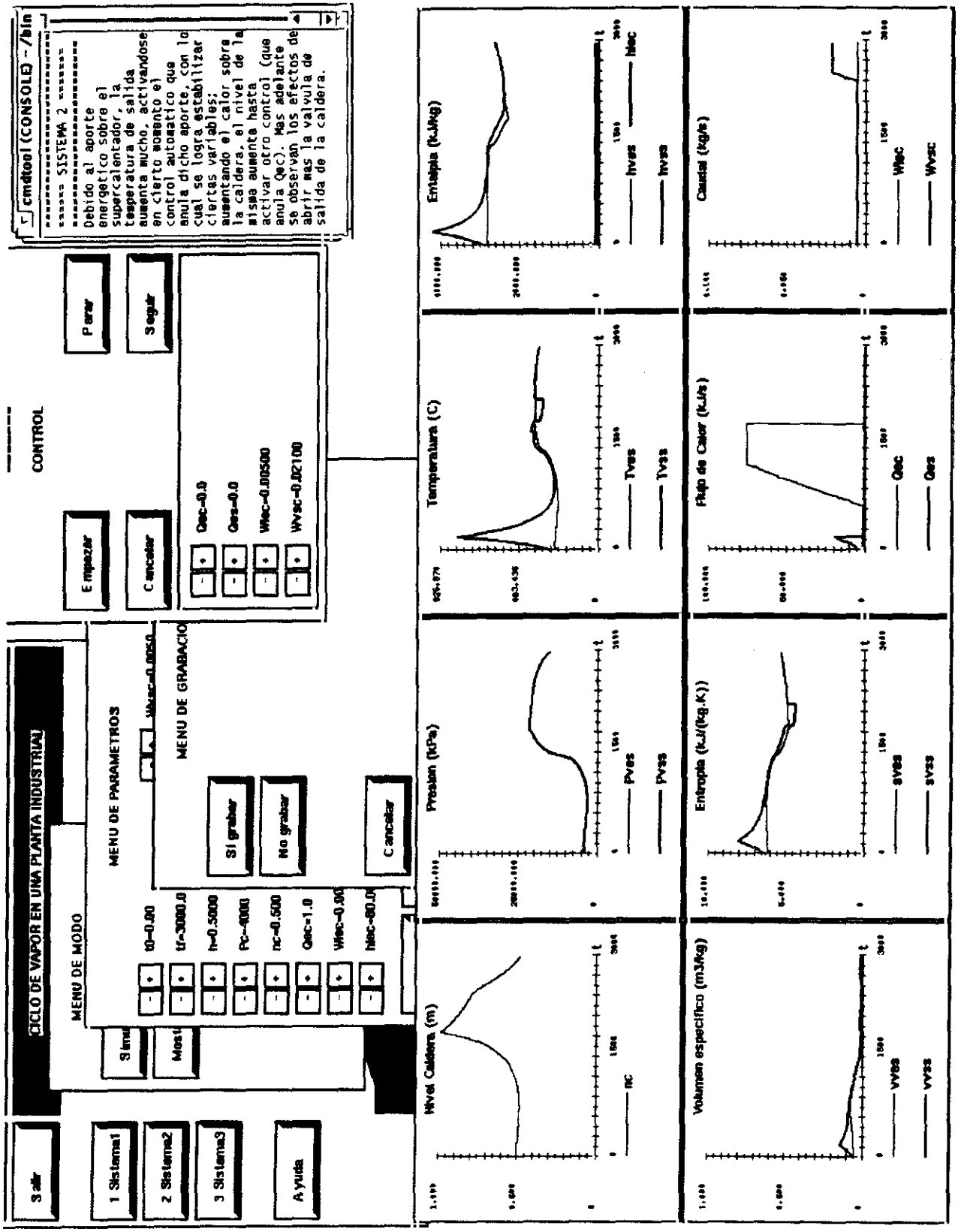

CICLO DE VAPOR EN UNA PLANTA INDUSTRIAL
MENU DE MODO
MENU DE PARAMETROS
MENU DE GRABACIO
CONTROL
Salir
1 Sistema1
2 Sistema2
3 Sistema3
Ayuda
Empezar
Cancelar
Parar
Seguir
Si grabar
No grabar
Cancelar
cmdtool (CONSOLE) - /bin
SISTEMA 2
Nivel Caldera (m)
Presion (kPa)
Temperatura (C)
Entalpia (kJ/kg)
Volumen especifico (m3/kg)
Entropia (kJ/(kg.K))
Flujo de Calor (kJ/s)
Caudal (kg/s)

Figura 6.21

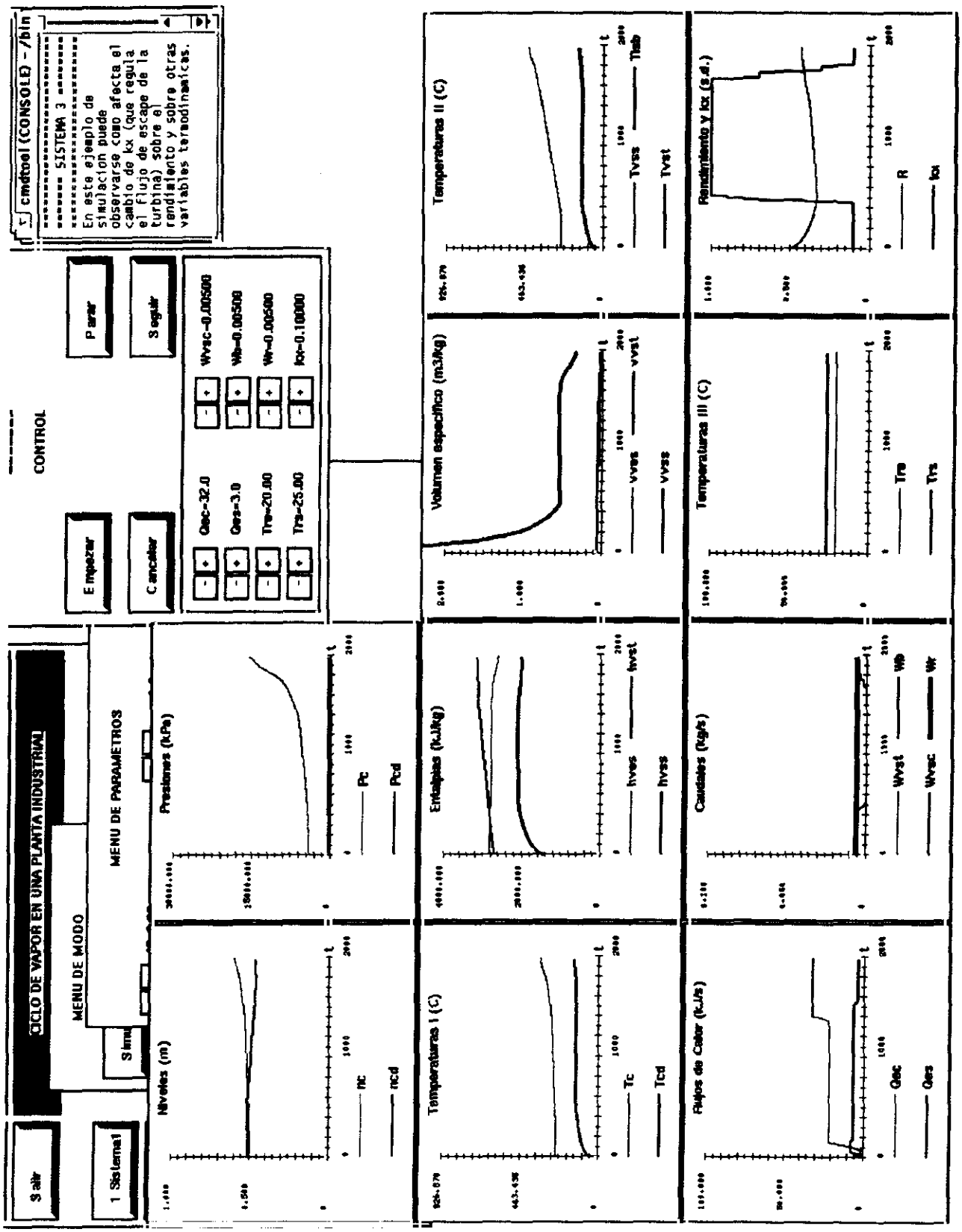

cmdtool (CONSOLE) - /bin
SISTEMA 3
En este ejemplo de simulacion puede observarse como afecta el cambio de kx (que regula el flujo de escape de la turbina) sobre el rendimiento y sobre otras variables termodinamicas.
CONTROL
Empezar
Cancelar
Parar
Seguir
Qsc=32.0
Qes=3.0
Tre=20.00
Trs=25.00
Wvsc=0.00500
Wb=0.00500
Ww=0.00500
kx=0.10000
CICLO DE VAPOR EN UNA PLANTA INDUSTRIAL
MENU DE MODO
MENU DE PARAMETROS
Salir
1 Sistema1
Niveles (m)
Presiones (kPa)
Volumen especifico (m3/kg)
Temperaturas II (C)
Temperaturas I (C)
Entalpias (kJ/kg)
Temperaturas III (C)
Rendimiento y kx (s.d.)
Flujos de Calor (kJ/s)
Caudales (kg/s)
nc
ncd
Pc
Pcd
Tc
Tcd
vvsc
vvst
vvss
hvsc
hvst
hvss
Tre
Trs
R
kx
Qsc
Qes
Wvst
Wb
Wvsc
Ww
Tsb
Tvsc
Tvst
Tvss

Figura 6.22

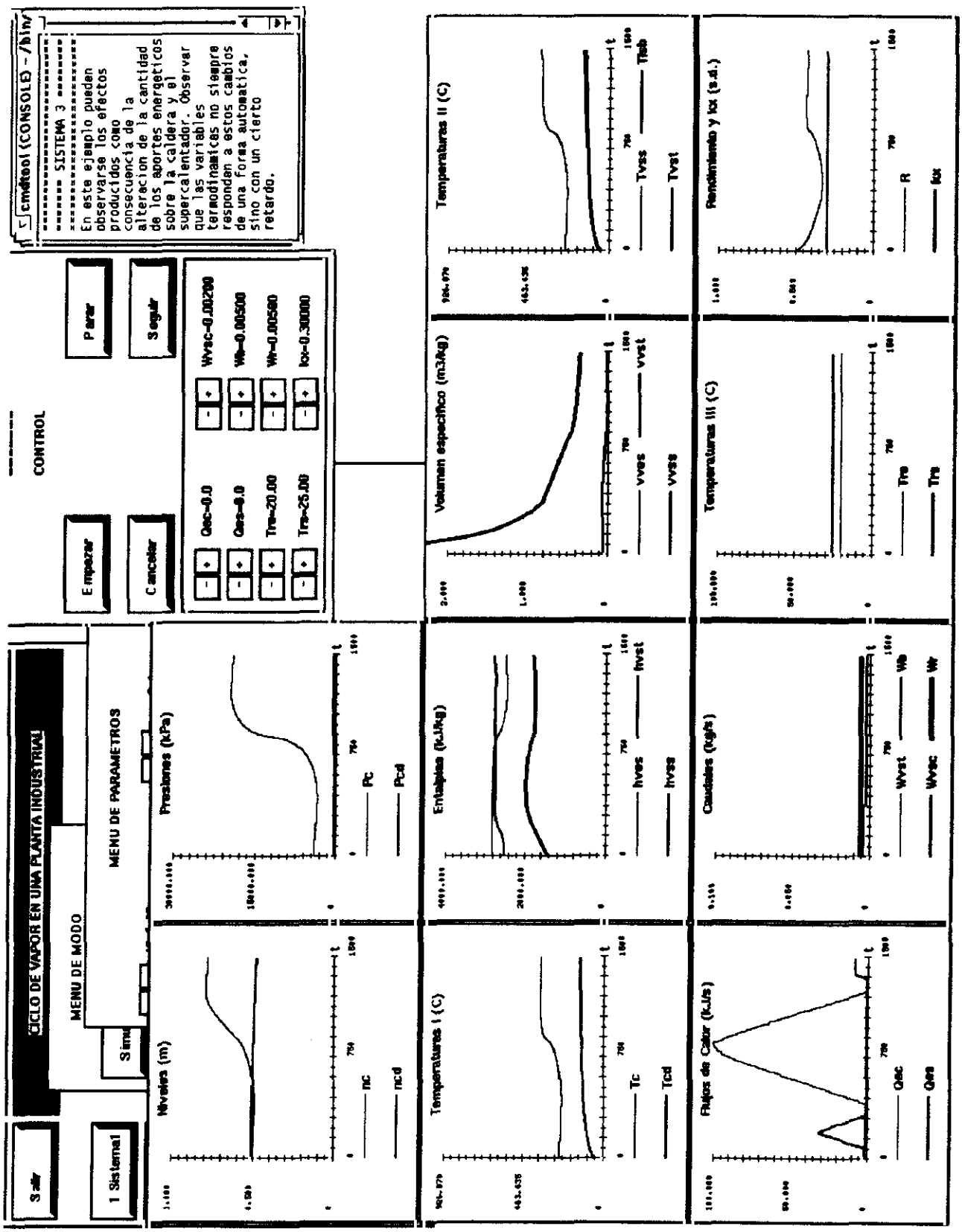

cmdtool (CONSOLE) - /bin/
En este ejemplo pueden observarse los efectos producidos como consecuencia de la alteracion de la cantidad de los aportes energeticos sobre la caldera y el supercalentador. Observar que las variables termodinamicas no siempre responden a estos cambios de una forma automatica, sino con un cierto retardo.
CONTROL
Parar
Seguir
Empezar
Cancelar
Qsc=0.0
Qes=0.0
Tre=20.00
Trs=25.00
Wvsc=0.00200
Wb=0.00500
Wr=0.00580
kx=0.30000
CICLO DE VAPOR EN UNA PLANTA INDUSTRIAL
MENU DE MODO
MENU DE PARAMETROS
Salir
1 Sistema
Niveles (m)
Presiones (kPa)
Temperaturas I (C)
Entalpias (kJ/kg)
Volumen especifico (m3/kg)
Temperaturas II (C)
Caudales (kg/s)
Temperaturas III (C)
Rendimiento y kx (p.d.)
Flujos de Calor (kJ/s)

Figura 6.23

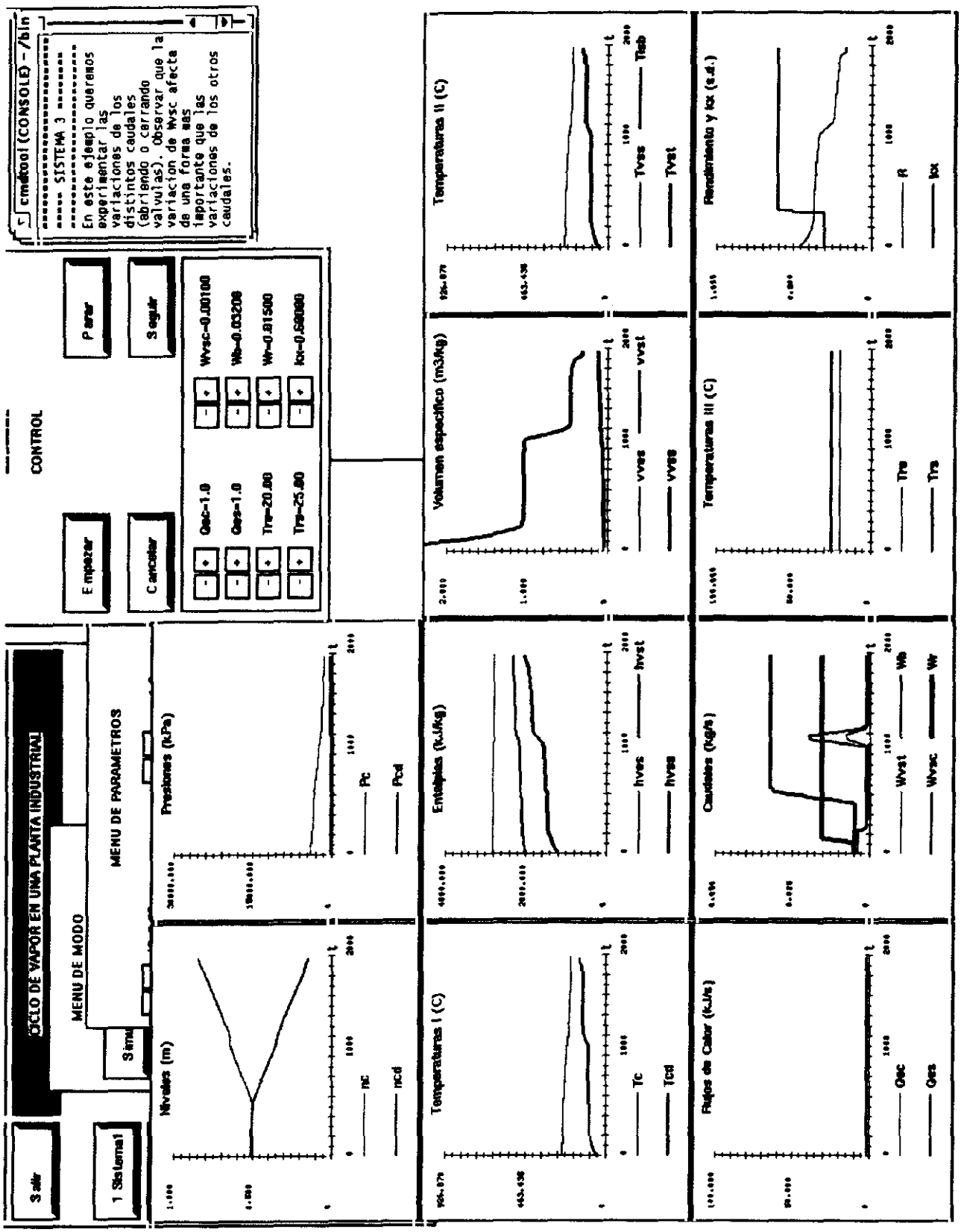

CICLO DE VAPOR EN UNA PLANTA INDUSTRIAL
Salir
1 Sistema1
MENU DE MODO
MENU DE PARAMETROS
Empezar
Cancelar
CONTROL
Parar
Seguir
Qsc=1.0
Qes=1.0
Tre=20.00
Trs=25.00
Wvsc=0.00100
Wb=0.03208
Wr=0.91500
kx=0.58000
cmdtool (CONSOLE) - /bin
SISTEMA 3
En este ejemplo queremos experimentar las variaciones de los distintos caudales (abriendo o cerrando valvulas). Observar que la variacion de Wvsc afecta de una forma mas importante que las variaciones de los otros caudales.
Niveles (m)
Presiones (kPa)
Temperaturas I (C)
Entalpias (kJ/kg)
Volumen especifico (m3/kg)
Temperaturas II (C)
Caudales (Kg/s)
Temperaturas III (C)
Rendimiento y kx (s.d.)
Flujos de Calor (kJ/s)

Figura 6.24

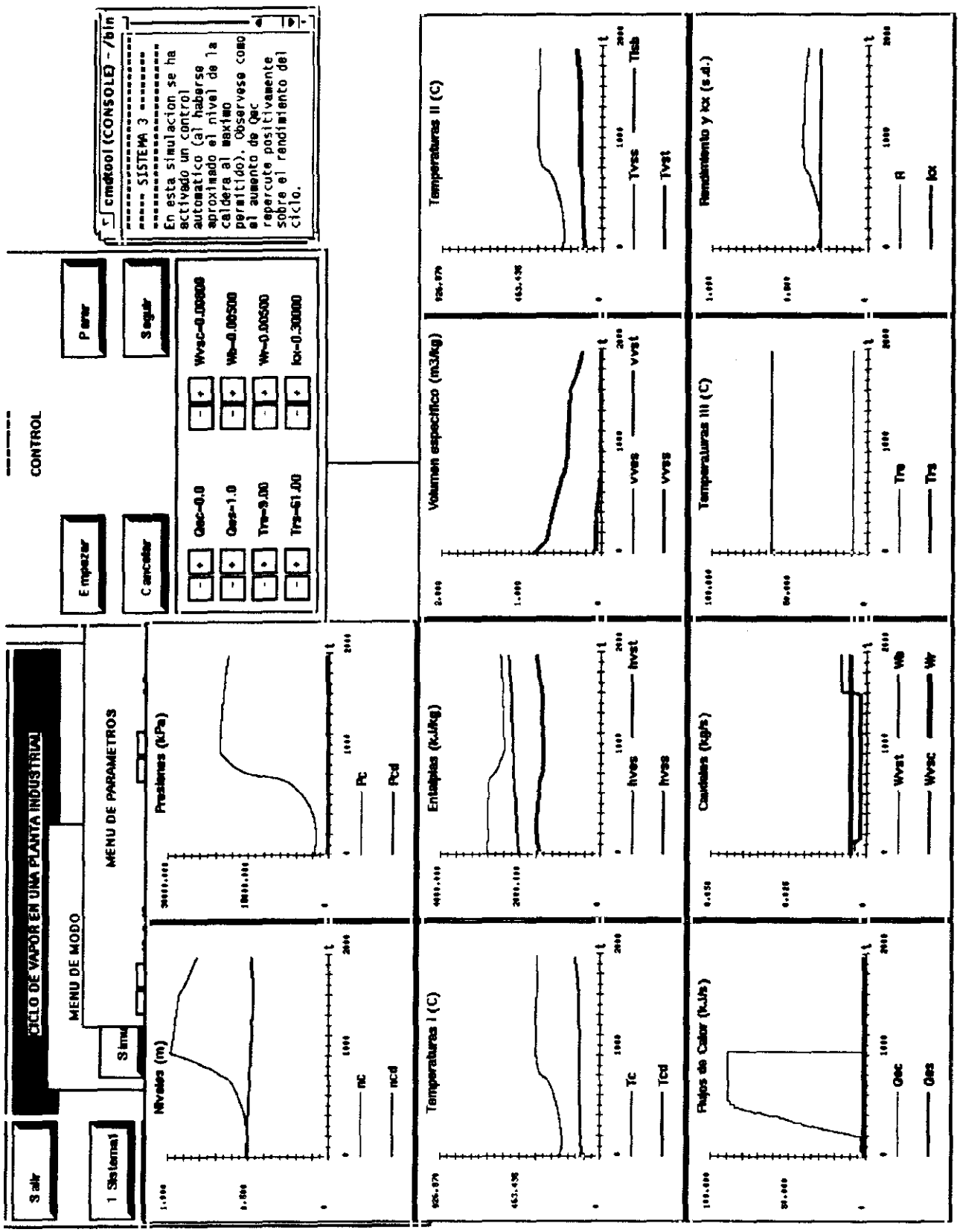

CICLO DE VAPOR EN UNA PLANTA INDUSTRIAL
MENU DE MODO
MENU DE PARAMETROS
Salir
1 Sistema1
Simul
Empezar
Cancelar
Parar
Seguir
CONTROL
Qac=0.0
Qas=1.0
Tre=9.00
Trs=61.00
Wvac=0.00000
Wb=0.00500
Wr=0.00500
lox=0.30000
cmdtool (CONSOLE) - /bin
SISTEMA 3
En esta simulacion se ha activado un control automatico (al haberse aproximado el nivel de la caldera al maximo permitido). Observese como el aumento de Qac repercute positivamente sobre el rendimiento del ciclo.
Niveles (m)
nc
ncd
Presiones (kPa)
Pc
Pcd
Temperaturas I (C)
Tc
Tcd
Entalpias (kJ/kg)
hvc
hvst
hvs
Volumen especifico (m3/kg)
vvc
vvst
vvs
Temperaturas II (C)
Tvc
Tvst
Tvs
Caudales (kg/s)
Wb
Wvst
Wvac
Wr
Temperaturas III (C)
Tre
Trs
Rendimiento y lox (s.d.)
R
lox
Flujos de Calor (kJ/s)
Qac
Qas

Figura 6.25

Conclusiones:

Conclusiones:

Exponemos en las páginas que siguen, brevemente, una síntesis de los resultados obtenidos por nuestra investigación, resaltando sus características más importantes.

La idea orientadora de nuestro trabajo ha sido contribuír, mediante la incorporación de tecnología actual, a nuevos métodos de estudio y entrenamiento basados en simulación, con respecto a las plantas térmicas de producción de electricidad. Concretamente, nuestro objetivo ha sido sacar partido de la potencialidad gráfica y de cálculo de las estaciones de trabajo, para ofrecer una alternativa a los métodos clásicos, basados en paneles mímicos de control. Nos interesa la enorme ductilidad de la pantalla gráfica del ordenador, que hace posible un empleo múltiple en cuanto a diversos tipos de visualización, también animada, y a la interacción con el usuario a través de ventanas y ratón.

Para llevar a cabo nuestro proyecto, hemos abordado la utilización de nuevas metodologías de programación (Programación Orientada a Objeto), y de interfases de usuario (X-Windows).

Acto seguido, procedemos a exponer ordenadamente los resultados alcanzados, comentando sus aspectos más relevantes.

Resultados Principales:

A lo largo de nuestro trabajo hemos desarrollado:

1 - Una modelación estática del ciclo de Rankine.-

Constituye un primer paso para alcanzar el objetivo final de la investigación, siendo ocasión de analizar en profundidad el campo de estudio abordado. Entre otras cosas, nos da información sobre los límites de ciertas magnitudes termodinámicas, niveles de rendimiento, etc.

2 - Una modelación dinámica.-

Se han obtenido las ecuaciones que modelan la dinámica de algunos dispositivos con los cuales poder construir un sistema que describa un ciclo termodinámico. Se ha efectuado bajo la metodología de la Programación Orientada a Objeto (POO).

3 - Una simulación dinámica del sistema.-

Con los modelos conseguidos, mediante POO y bajo entornos gráficos X- Windows sobre una estación de trabajo, se desarrolló una aplicación que efectúa simulaciones del ciclo considerado bajo diversas condiciones.

Marco de Referencia:

Para realizar estas tareas, hemos tenido que estudiar previamente dos aspectos para establecer la oportunidad de nuestra investigación :

1 - Análisis bibliográfico sobre la modelación.-

A la vista de los trabajos consultados sobre los modelos de plantas de vapor, podemos extraer las siguientes conclusiones:

— Hay modelos globales de la planta; tienen la desventaja de no poder aprovechar informaciones parciales para aplicarlas sobre otros sistemas. Además, estos modelos suelen ser estáticos, careciendo de gran interés.

— Otros modelos (los más numerosos) se basan en sub-sistemas o dispositivos. Dentro de la amplia variedad que se ofrece, dividimos a grandes rasgos los que se basan en estudios teóricos (que consideran la física de los procesos), y los que son resultado del tratamiento de resultados experimentales (como las identificaciones en plantas reales).

— La mayoría de los modelos están descritos en variables de estado temporales (lo más adecuado), aunque se encuentran también formulaciones en la transformada "s" (bajo la perspectiva del control).

— Prácticamente la mayoría de estos modelos son poco utilizables por dos motivos: por su "localismo" (están basados en dispositivos reales que trabajan bajo condiciones concretas) y por bajo nivel de explicación (las empresas son reacias a dar los detalles vitales, de interés para la competencia).

2 - Análisis bibliográfico sobre la simulación.-

Estudiando las simulaciones publicadas, llegamos a las siguientes conclusiones :

— Bastantes son poco llamativas desde el punto de vista gráfico. Además, muchas de las simulaciones tradicionales se basan en paneles mímicos. Todo ello creemos que puede suponer dificultades para los que carecen de suficientes conocimientos iniciales, restando su capacidad de experimentación.

— Están programadas en lenguajes convencionales (BASIC y FORTRAN) o mediante paquetes de simulación específicos (MMS, etc.), con posibles problemas de portabilidad o de precio. Algunos de estos aspectos están en fase de superación ante la entrada de nuevas técnicas y su progresivo abaratamiento.

Aportaciones Realizadas:

Los aspectos más importantes de nuestro trabajo, y que consideramos constituyen una aportación frente a la anteriormente descrito en las conclusiones de la revisión bibliográfica son :

Respecto a la modelación :

— Centramos nuestro interés sobre la física de los procesos, aumentando el nivel didáctico en la exposición de los modelos, con objeto de acercarlos a etapas conceptuales de estudio o entrenamiento.

— Nos interesan los fenómenos que experimenta el agua a lo largo del ciclo termodinámico. Dejamos para posibles adaptaciones a ejemplares industriales de plantas, la incorporación de efectos derivados de cuestiones tecnológicas: materiales, formas, etc.

— Seguimos un criterio de modularidad, preconizada por la POO, que facilita la utilización de modelos parciales (de componentes) para la configuración de sistemas distintos, a la manera de un mecano.

— Hemos creado unos arrays (utilizables como base de datos para cualquier aplicación programada en C) que detallan las tablas de vapor, ofreciendo una ingente información del todo necesaria para los requisitos de cualquier investigación en el ámbito termodinámico. También hemos desarrollado unas funciones que gestionan estas tablas tanto en la búsqueda como en la interpolación de datos.

Respecto a la simulación :

— Incorporación de un lenguaje de grandes posibilidades (C++) en el ámbito de las estaciones de trabajo, con técnicas POO, lo cual ayuda a una mayor claridad en el código del programa y a una gran transportabilidad tanto entre máquinas como entre aplicaciones.

— Dotación de una buena visualización gráfica e interactividad con el usuario mediante el moderno entorno X-Windows. En cuanto al alto grado de manejabilidad de la aplicación, se consigue que el usuario pueda con entera libertad interaccionar con teclado o ratón abriendo ventanas, desplegando menús, oprimiendo botones, etc.; en cuanto al impacto visual, se alcanza una presentación agradable y vistosa. El uso de X-Windows también logra amparar a nuestra aplicación con unos estándars y normalizaciones que la hacen sumamente transportable a diversos tipos de ordenadores, y con la ventaja

de su uso en red. Hemos de destacar que en la programación en X- Windows hemos utilizado directamente sus librerías.

— Uso de máquinas superiores en velocidad de procesamiento a cualquier PC actual. Destacamos que la aplicación desarrollada para la estación Sun Sparc 1+, trabaja tanto en el entorno X-Windows como en el originario Sun-OpenWindows.

— También pueden encontrarse pequeñas aportaciones de utilidad como la gestión de ficheros para el almacenamiento y recuperación de simulaciones, técnicas de programación con el Scheduler, etc.

Características de las Aplicaciones Creadas:

Las aportaciones mencionadas surgen en las distintas etapas de la investigación, marcadas por la creación de sucesivas aplicaciones, cada vez más avanzadas. Por este motivo, pondremos de relieve sus características en relación a las diversas fases de nuestro trabajo:

1 - Introducción con la aplicación RANKINE.EXE :

— Simulación del ciclo de Rankine en su aspecto estático, ofreciendo diversas posibilidades de experimentación y dando estados y rendimientos como resultados, con presentaciones gráficas.

— Uso del lenguaje C, sobre ordenador personal PC.

— Creación de los arrays y funciones que implementan y gestionan las tablas de vapor.

2 - Modelación Dinámica:

— Análisis de funciones y estructura de una planta térmica basada en ciclo de Rankine. Taxonomía de clases y objetos que permiten una modelación modular.

— Modelos individuales de los dispositivos, expresados mediante POO.

— Descripción dinámica en variables de estado, abundando en su explicación física.

— Elaboración propia de modelos concretos (condensador) no encontrados en la bibliografía.

3 - Simulación (aplicacion PLANTA en dos versiones) :

— Creación de tres ejemplos de sistemas térmicos, en sucesivos niveles de complejidad, que permiten visualizar comportamientos de interés de las plantas térmicas. El tercer ejemplo constituye el ciclo completo de un prototipo de planta de vapor.

— Creación de un Scheduler para la organización del código en el reparto de tareas. Definido también según las directivas de la POO.

— Programación bajo entorno X-Windows (aplicación PLANTA).

— Uso de estaciones de trabajo (Sun Sparc 1+) para la aplicación PLANTA, y PC's para la aplicación PLANTA.EXE.

— Dotación de control automático en el software.

— Interacción de usuario mediante ratón y teclado, menús desplegables, botones, ventanas, gráficas, etc. (aplicación PLANTA).

— Posibilidad de almacenamiento y recuperación de simulaciones.

— Amplios márgenes de experimentación, mediante la selección de los valores iniciales de numerosas variables, y con la actuación en tiempo de simulación de variables de control.

— Ofrecimiento de ayudas y explicaciones.

En el Capítulo VII incluímos algunos ejemplos de simulación, presentando la forma en que visualizamos los resultados mediante un conjunto de gráficos. Estos gráficos aparecen en pantalla de forma animada, reflejando el decurso de los acontecimientos en la planta.

Detalles Técnicos:

1 - Tamaños de las aplicaciones :

Consignamos el número de caracteres de las aplicaciones, según su listado en fuente C o C++:

— RANKINE:

109.518 (para Turbo-C) (aprox. 1800 líneas)

— PLANTA.EXE:

116.434 (para Borland C++) (aprox. 1900 líneas)

— PLANTA:

234.394 (en GNU C++ sobre SUN) (aprox. 3900 líneas)

2 - Tiempos de ejecución :

— RANKINE.EXE:

Al no tratarse de simulaciones dinámicas, el cálculo es inmediato. Por otra parte, es previsible que los usuarios en primeras etapas de aprendizaje, dedicarán más tiempo al estudio de las pantallas explicativas que a las experimentaciones.

— PLANTA.EXE:

Los tiempos de cálculo asociados a la simulación de los sistemas dinámicos dependerán, obviamente, del tiempo total, amplitud de paso, etc. que seleccionemos en los menús de parámetros iniciales. Además, al tener que ejecutarse sobre un PC, dependerá tanto del tipo y velocidad del procesador como de la existencia o no de un coprocesador matemático. Con todo, los tiempos serán más largos que los logrados por una estación de trabajo.

— PLANTA:

Al ejecutarse sobre estaciones de trabajo, los tiempos de cálculo son muy satisfactorios, según detallamos en el Capítulo VII.

3 - Horas de desarrollo estimadas:

- Etapas de estudio y análisis del problema: 300
- Escritura de tablas de vapor en arrays: 100
- Desarrollo de RANKINE.EXE: 300
- Elaboración de los modelos dinámicos: 350
- Desarrollo de PLANTA.EXE : 180
- Aprendizaje en el manejo de estaciones de trabajo: 40
- Instalación de GCC y X-Windows en una estación Sun-Sparc: 40
- Estudio de la programación en X-Windows: 90
- Desarrollo de PLANTA: 300

Comentario de los Resultados. Posibles Líneas de Ampliación e Investigación Futura:

Examinando el conjunto de resultados obtenidos, nos parece conveniente señalar, siempre bajo nuestro punto de vista, su alcance y limitaciones.

Respecto a la modelación, se puede decir que es correcta en cuanto no ofrece contradicciones con la teoría termodinámica. Sin embargo, es de destacar que no se trata de un sofisticado modelo correspondiente a un sistema real. Por otra parte, hemos adoptado una serie de supuestos que delimitan una zona de valores en las que se mueve el modelo (excluyendo situaciones muy especiales, sólo alcanzables en incidentes destructivos); zona que es suficientemente válida para los propósitos generales que nos hemos impuesto en el estudio de la dinámica del sistema.

La delimitación de una zona se concreta en lo relativo a las tablas de vapor, organizadas mediante arrays, de modo que no cubran la totalidad de los estados teóricos del agua, aunque sí los necesarios para los límites de operación normales; por tanto, si en el transcurso de una simulación se superasen las zonas consideradas por las tablas, se producirían cálculos incorrectos; para evitar ésto y tener un amplio margen de experimentación, se han incorporado controles automáticos en el software y se ofrecen variables de control en tiempo de simulación.

La oportunidad o no en el empleo de las técnicas de POO con C++ y el uso de X-Windows se demuestra, para el primer caso, en la limpieza y orden del código, su transportabilidad y re-usabilidad, aparte de las numerosas ventajas propias de las librerías C (conexión directa con el lenguaje en el que está programado el sistema operativo de la estación de trabajo); y para el segundo caso, en el impacto visual y alto nivel de interactividad que proporciona dicho entorno Windows.

En buena medida, pretendíamos evaluar, con las aplicaciones que hemos desarrollado, las posibilidades y ventajas reales de la POO en simulaciones de cierta envergadura. Dentro del nivel de complejidad alcanzado por nuestro trabajo, hemos podido comprobar que efectivamente se tienen las ventajas prometidas por la POO y por X-Windows. Se temía que tuviera repercusiones sobre el tiempo de ejecución, pero afortunadamente es muy satisfactorio sobre la base de una estación de trabajo convencional, como es la que hemos tenido ocasión de utilizar.

De acuerdo con los comentarios que acabamos de realizar, están a la vista algunas líneas de continuación de las investigaciones: pasar a la aplicación a ejemplos reales de plantas térmicas, ampliar la zona de trabajo, evaluar posibles fuentes de error y sus implicaciones (por ejemplo, el error inherente a las interpolaciones y las discontinuidades podría tener efectos acumulativos en ciertas circunstancias), mejorar aspectos estéticos y de ergonomía en las visualizaciones y menús, realizar estudios de aplicación para entrenamiento, puesta a punto para uso profesional, etc.

Desde la perspectiva académica, nos interesa añadir características inteligentes a la simulación, de modo que detecte situaciones que merezcan un comentario específico en pantalla.

APENDICES

Apéndice I:

REFERENCIAS Y BIBLIOGRAFIA

Apéndice I:

REFERENCIAS Y BIBLIOGRAFIA

A1.1.- LIBROS DE INGENIERIA TERMODINAMICA.-

- **Howell, J & Buckius, Richard (1987)** "Fundamentals of Engineering Thermodynamics" McGraw-Hill, New York

- **Joel, Rayner (1987)** "Basic Engineering Thermodynamics" Longman, UK

A1.2.- LIBROS DE MODELACION Y SIMULACION DE PLANTAS DE VAPOR.-

- **Knowles, J.B. (1990)** "Simulation and Control of Electrical Power Stations" Research Studies Press Ltd. John Wiley & Sons.

- **Polonyi, Michael (1991)** "Power and Process Control Systems" McGraw-Hill, New York.

☐ **Weisman, Joel & Eckart,L.E. (1985)** "Modern Power Plant Engineering" Prentice-Hall, NJ.

A1.3.- TESIS DOCTORALES SOBRE LA MODELACION Y SIMULACION DE PLANTAS DE VAPOR.-

☐ **Azuma, Alberto (1975)** "Modeling and simulation of a steam power station" Massachusetts Institute of Technology

☐ **Masada, Glenn (1979)** "Modeling and control of power plant boiler- turbine-generator systems" Massachusetts Institute of Technology

☐ **Usoro, Patrick Benedict (1977)** "Modeling and simulation of a drum boiler-turbine power plant under emergency state control" Massachusetts Institute of Technology

A1.4.- ARTICULOS SOBRE LA MODELACION Y SIMULACION DE SISTEMAS TERMICOS.-

☐ **Babula, Dusan (1986)** "Modelling the primary system of a PIUS nuclear power plant" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

☐ **Barcelo, W.R. et al (1985)** "Modeling of the MSUS standard coal-fired unit" Modular Modeling System. Proceedings. Electric Power Research Institute.

☐ **Belblidia, L.A. et al (1986)** "A lumped-parameter steam generator model for dynamic simulator for nuclear power plants (DSNP) " "A level-two pressurizer model for DSNP" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

☐ **Berenbach, Brian et al. (1991)** "The design and implementation of an object-oriented power plant training simulator" SIMULATORS VIII. Volume 24, Nº 1. SCS, 1991.

☐ **Bernard, J. et al (1986)** "The application of digital technology to the control of reactor power: a review of the M.I.T. reactor experiments" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

- **Chaplin, R.A. & Surgenor, B.W. (1986)** "Modelling of a power plant feedwater system for the development of a disturbance accommodating controller" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

- **Dieck-Assad, Graciano (1990)** "Development of a state space boiler model for process optimization" SIMULATION, October 1990.

- **Dunne, James et al. (1984)** "Fossil-fueled power plant steady-state simulator model" International Conference on Power Plant Simulation, 1984.

- **Kanodia, H.K. et al. (1988)** "PC based training systems for power and process plants" Power Plant Simulation, 1988. SCS International.

- **Khadem, M. et al. (1986)** "A compact, interactive and color-graphics based simulator for power plant analysis" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

- **Kubiak, Janusz (1984)** "Modeling of turbines for analysis of control systems" International Conference on Power Plant Simulation, 1984.

- **Kwatny, H.G. & Bauerle, J.W. (1986)** "Modelling of low load dynamics of coal fired furnaces" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

- **Li-Chi Cliff Po, (1986)** "PCTRAN/B - A "personal" BWR transient simulator" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

- **Ocampo, H. & González, J.L. (1984)** "Turbine model used in a simulator of a fossil power plant" International Conference on Power Plant Simulation, 1984.

- **Rajakumar, A. & Vaidyanathan, G. (1986)** "Simulation and control studies for FBTR steam/water system" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

- **Riley, K. (1984)** "The design and implementation of the KRAB steady- state total plant modelling system and the on-line on-site applications of its off spring" International Conference on Power Plant Simulation, 1984.

- **Roldán-Villasana, E. & Del-Arco, Carlos (1984)** "Model of the flows and pressure in the water circuit of a forced circulation boiler" International Conference on Power Plant Simulation, 1984.

- **Rosard, D. (1985)** "Use of computer models to study cyclic operation" Modular Modeling System. Proceedings. Electric Power Research Institute.

- **Saphier, D. & Belblidia, L.A. (1986)** "A higher-level simulation language, the DSNP, and some of its applications" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

- **Smith, Lance P. (1985)** "Description of fossil and balance of plant component models in the modular modeling system" Modular Modeling System. Proceedings. Electric Power Research Institute.

- **Stegemann, D. et al (1986)** "Computer models to simulate PWR nuclear power plants" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

- **Sukanit, J. et al. (1986)** "Real-time simulation of a nuclear u-tube steam generator for operations and contingency analysis" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

- **Zwingelstein, G. & Despujols, A. (1986)** "SINCRO/CAR: An interactive numerical system for computer-aided control engineering and maintenance" 6th. Power Plant Dynamics, Control & Testing Symposium, 1986.

A1.5.- TABLAS TERMODINAMICAS DE VAPOR.-

- **Grigull, U. - Straub, Johannes - Schiebener, Peter (1990)** "Steam Tables in SI-Units" Springer-Verlag. Berlin.

- **Reynolds, W. C. (1979)** "Thermodynamic Propierties in S.I. Graphs, Tables and Computational Equations for 40 Substances" Department of Mechanical Engineering, Standford University.

A1.6.- PROGRAMACION ORIENTADA A OBJETO Y C++.-

- **Voss, Gregg and Chui, Paul (1990)** "Turbo C++ Disk Tutor" Borland-Osborne/McGraw-Hill, Berkeley.
- **"Turbo C++ Getting Started" (1990)** Borland International.
- **"Turbo C++ User's and Programmer's Guides" (1990)** Borland International.
- **Girón Sierra, J.M. (1991)** "Introducción a la programación orientada a objeto" Seminario. Universidad Complutense de Madrid.

A1.7.- PROGRAMACION C.-

- **Zimmerman, S & Zimmerman, B.B. (1989)** "La Biblia del Turbo C" Anaya Multimedia, Madrid.
- **Weiskamp, Keith - Shammas, Namir - Pronk, Ron (1989)** "Turbo Libraries. A Programmer's Reference" John Wiley & Sons, Inc. New York.
- **"Turbo C : User's Guide and Programmers's Guide"** Borland International.
- **"Science & Engineering Tools for Turbo C" (1988)** Quinn-Curtis, MA. USA.

A1.8.- PROGRAMACION EN X-WINDOWS.-

- **Barkakati, Naba (1991)** "X Window System Programming" SAMS - MacMillan Computer Publishing.
- **Johnson, Eric F. & Reichard, Kevin (1989)** "X Window Applications Programming" Advanced Computer Books. MIS Press.

Apéndice II:

LISTADO DEL PROGRAMA RANKINE.EXE

Apéndice II:

LISTADO DEL PROGRAMA RANKINE.EXE

Características:

Desarrollado en Turbo-C.

Corre en plataformas MS-DOS (preferible 486 con VGA color).

Se dedica a una simulación estática para introducciones conceptuales.

Incluye las tablas termodinámicas del agua.

PROGRAMA RANKINE.EXE

RANK.H

```
#include "c:\tc\include\graphics.h"                    /* ficheros cabecera */
#include "c:\tc\include\conio.h"
#include "c:\tc\include\math.h"
#include "c:\tc\include\stdio.h"
#include "c:\tc\include\stdlib.h"
#include "c:\tc\include\ctype.h"

#define PI 3.1415927

void present(void);                                    /* declaraciones de funciones */
void dat_rank(int noc);
void menu(void);
void menu_rank(int noc);
void res_num_is(void);
void res_num_rs(void);
void res_num_sup(void);
void res_num_rec(void);
void abrirgraf(void);
void rankis(void);
void rankrs(void);
void ranksup(void);
void rankrec(void);
void graf(int noc);
void graf_rend_rank(int noc);
void calc(int noc,double Ta,double Tb,double Rt,double Rp,double Tsc,double Ph);
void liqsat(double Temp);
void vapsat(double Temp);

void vapsupercal(double Temp,double Pres);
void vapsupercal2(double Pres,double entrop);
void copimag(int x1or,int y1or,int x2or,int y2or,int xcop,int ycop,int op);
double intrplcn(double,double,double,double,double);
void graf_resultados(double *ph,int noc);
int Max(double *pdat,int numdat);
void exp_is(void);
void exp_rs(void);
void exp_sup(void);
void exp_rec(void);
void exp_gen(void);
void esquema(int num);
void diagram(int num);
double vapsat_return_h(double Temp);
void error_message(void);
void textrankis1(void);
void textrankis2(void);
void textranksup(void);
void textrankrec(void);
void textrankrs(void);
```

RANKIS.C

```
#include "rank.h"

char c, tecla1;
double Ta, Tb, Rt, Rp, Tsc, Ph;
double x[12], T[12], P[12], v[12], s[12], h[12];
int x1, y1, error_signal;

void main()
{
present(); clrscr();
do  {
    menu();
    tecla1=toupper(getch());
```

```
    switch(tecla1)
        {
        case 'A': rankis(); break;
        case 'B': rankrs(); break;
        case 'C': ranksup(); break;
        case 'D': rankrec(); break;
        case 'X': textmode(C80); clrscr(); exit(1);
        default: printf("\a");
        }
    }
    while(tecla1 !='X');
}
/*=========================================*/
void rankis(void)
{
char tecla2;

do  {
    dat_rank(1);
    calc(1,Ta,Tb,Rt,Rp,Tsc,Ph);
    do  {
        menu_rank(1);
        tecla2=toupper(getch());
        switch(tecla2)
            {
            case 'D': clrscr(); res_num_is(); break;
            case 'G': clrscr(); graf(1); break;
            case 'E': clrscr(); exp_is(); break;
            case 'M': clrscr(); break;
            default: printf("\a");
            }
        }
        while(tecla2 !='M');
    }
    while(tecla2 !='M');
}
/*=========================================*/
void menu(void)
{
textmode(C80); clrscr();
textattr(14+0);
gotoxy(21,1); cputs("==================");
gotoxy(21,2); cputs("======= MENU ========");
gotoxy(21,3); cputs("==================");
textattr(15+3);
gotoxy(21,7) ;cputs("A === Ideal and Simple Rankine Cycle ");
gotoxy(21,9); cputs("B === Real and Simple Rankine Cycle");
gotoxy(21,11); cputs("C === Superheating in Rankine Cycle");
gotoxy(21,13); cputs("D === Reheating in Rankine Cycle");
gotoxy(21,19); cputs("X === Exit");
gotoxy(30,25); cputs("Type your choose option");
}
/*=========================================*/
void res_num_is(void)
{
textmode(C80); clrscr(); textattr(14+0);
gotoxy(5,2); cputs("STATES:");
gotoxy(5,3); cputs("=======");
gotoxy(10,5); cputs("x");
gotoxy(15,5); cputs("T (C)");
gotoxy(28,5); cputs("P (kPa)");
gotoxy(41,5); cputs("v (m3/kg)");
gotoxy(54,5); cputs("s (kJ/(kgK))");
gotoxy(67,5); cputs("h (kJ/kg)");
gotoxy(10,6); cputs("-----------------------------------------");
textattr(15+0);
gotoxy(5,9); cprintf("1% -9.3f %-6.4f %9.4f",P[1],s[1],h[1]);
gotoxy(5,11); cprintf("2 %-3.1f %-6.2f %-9.3f %-10.6f %6.4f %9.4f", x[2],T[2],P[2],v[2],s[2],h[2]);
gotoxy(5,13); cprintf("3 %-3.1f %-6.2f %-9.3f %-10.6f %-6.4f %-9.4f", x[3],T[3],P[3],v[3],s[3],h[3]);
gotoxy(5,15); cprintf("4 %-4.2f %-6.2f %-9.3f %-10.6f %-6.4f %9.4f", x[4],T[4],P[4],v[4],s[4],h[4]);
gotoxy(5,17); cprintf("5 %-3.1f %-6.2f %-9.3f %-10.6f %-6.4f %-9.4f", x[5],T[5],P[5],v[5],s[5],h[5]);
getch(); clrscr(); textattr(14+0);
gotoxy(25,5); cputs("  W (3-4) +W (5-1)");
gotoxy(25,6); cputs("s,s = - ----------");
gotoxy(25,7); cputs("  Q(1-3) ");
gotoxy(20,12); cputs("W(3-4)=h[4]-h[3]=");
textattr(15+0); cprintf("%f kJ/kg",h[4]-h[3]);
gotoxy(20,14); textattr(14+0); cputs("W(5-1)=h[1]-h[5])=");
textattr(15+0); cprintf("%f kJ/kg",v[5]*(P[1]-P[5]));
gotoxy(20,16); textattr(14+0); cputs("Q(1-2)=h[2]-h[1]=");
textattr(15+0); cprintf("%f kJ/kg",h[2]-h[1]);
gotoxy(20,18); textattr(14+0); cputs("Q(2-3)=h[3]-h[2]=");
textattr(15+0); cprintf("%f kJ/kg",(T[2]+273.15)*(s[3]-s[2]));
textattr(14+0); gotoxy(20,21); cputs("==================");
gotoxy(20,22); cputs("EFFICIENCY :s,s = ");
textattr(15+0);cprintf("%-10.4f%",-(h[4]+h[1]-h[3]-h[5])*100./fabs(h[3]-h[1]));
textattr(14+0); gotoxy(20,23); cputs("==================");
getch(); textmode(C80); clrscr();
graf_resultados(h,1);
}
/*=========================================*/
void present(void)
{
extern int xl, yl;
```

```
int ux, uy;

clrscr(); abrirgraf();
ux=(xl+1)/8; uy=(yl+1)/8;
rectangle(ux/2,uy/2,7.5*ux,7.5*uy);rectangle(ux,uy,7*ux,7*uy);
setfillstyle(1,9); floodfill(2,2,15);
setfillstyle(6,14); floodfill(ux-2,uy-2,15);
setfillstyle(1,0); floodfill(ux+2,uy+2,15);
setcolor(9); settextstyle(4,0,4); outtextxy(ux+20,2*uy," RANKINE CYCLES ");
settextstyle(0,0,1); outtextxy(2*ux,6*uy,"Hercules or EGAVGA card needed");
getch(); cleardevice(); closegraph();
}
/*=========================================*/
void abrirgraf(void)
{
int gd, gm, msgnum;
char *errstr;

detectgraph(&gd,&gm);
if(gd==-2)
    { printf("\aThere is not installed graphic driver"); getch(); exit(1); }
msgnum=graphresult();
if(msgnum != 0)
    { errstr=grapherrormsg(msgnum); printf("\aGraphic error detected");
      printf("\nThe error code is %d:%s",msgnum,errstr); getch(); exit(1); }
if(registerbgidriver(EGAVGA_driver))                    exit(1);
if(registerbgidriver(ATT_driver))                       exit(1);
if(registerbgidriver(CGA_driver))                       exit(1);
if(registerbgidriver(Herc_driver))                      exit(1);
if(registerbgifont(small_font))                         exit(1);
if(registerbgifont(gothic_font))                        exit(1);
if(registerbgifont(triplex_font))                       exit(1);
if(registerbgifont(sansserif_font))                     exit(1);
initgraph(&gd,&gm,"c:\\tc");
graphdefaults();
xl=getmaxx(); yl=getmaxy();
}
/*=========================================*/
void copimag(int x1or,int y1or,int x2or,int y2or,int xcop,int ycop,int op)
{
unsigned tamrect;
void *bufimag;

tamrect=imagesize(x1or,y1or,x2or,y2or);
bufimag=malloc(tamrect);
getimage(x1or,y1or,x2or,y2or,bufimag);
putimage(xcop,ycop,bufimag,op);
free(bufimag);
}
/*=========================================*/
void graf_resultados(double *ph,int noc)
{
int a, as, ar, a0=0;/* ángulos */
double R, Rs, Ra; /* Rendimiento */
double dat[20];
int ux, uy, Energ_max,maxy, top;
char read[30];

ux=(xl+1)/16; uy=(yl+1)/12;
maxy=5*uy;

abrirgraf();
rectangle(0.4*ux,0.4*uy,15.6*ux,11.6*uy);
setfillstyle(1,LIGHTBLUE); floodfill(2,2,WHITE);
settextstyle(SMALL_FONT,HORIZ_DIR,4);
switch(noc)
    {
    case 1: dat[0]=fabs(ph[2]-ph[1]);                   dat[1]=fabs(ph[3]-ph[2]);
            dat[2]=fabs(ph[1]-ph[5]);                   dat[3]=fabs(ph[4]-ph[3]);
            dat[4]=fabs(ph[5]-ph[4]);
            Energ_max=Max(dat,5);
            R=-100.*(ph[4]+ph[1]-ph[3]-ph[5])/fabs(ph[3]-ph[1]);
            setfillstyle(2,RED);top=10*uy-maxy*dat[0]/Energ_max;
            bar(3*ux,top,4*ux,10*uy); outtextxy(3*ux,top-uy,"|Q(1-2)|"); /*Q12*/
            setfillstyle(6,RED); top=10*uy-maxy*dat[1]/Energ_max;
            bar(4*ux,top,5*ux,10*uy); outtextxy(4*ux,top-uy,"|Q(2-3)|"); /*Q23*/
            setfillstyle(4,RED); top=10*uy-maxy*dat[2]/Energ_max;
            bar(5*ux,top,6*ux,10*uy); outtextxy(5*ux,top-uy,"|W(5-1)|"); /*W51*/
            setfillstyle(1,LIGHTBLUE);top=10*uy-maxy*dat[3]/Energ_max;
            bar(8*ux,top,9*ux,10*uy); outtextxy(8*ux,top-uy,"|W(3-4)|"); /*W34*/
            setfillstyle(9,GREEN);top=10*uy-maxy*dat[4]/Energ_max;
            bar(11*ux,top,12*ux,10*uy);outtextxy(11*ux,top-uy,"|Q(4-5)|");/*Q45*/
            settextstyle(SMALL_FONT,HORIZ_DIR,5);
            outtextxy(3*ux,10.5*uy,"SPENT ENERGY");
            outtextxy(8*ux,10.5*uy,"GIVEN ENERGY");
            outtextxy(11*ux,10.5*uy,"LOST ENERGY");
            outtextxy(6*ux,1.5*uy,"          W(3-4)+W(5-1)");
            outtextxy(6*ux,2*uy,"Efficiency = - ---------- =");
            outtextxy(6*ux,2.5*uy,"          |Q(1-2)+Q(2-3)|");
            sprintf(read,"%-5.2f%",R);outtextxy(13.5*ux,2*uy,read);
            break;
    case 2: dat[0]=fabs(ph[3]-ph[2]);                   dat[1]=fabs(ph[4]-ph[3]);
            dat[2]=fabs(ph[2]-ph[7]);                   dat[3]=fabs(ph[6]-ph[4]);
            dat[4]=fabs(ph[7]-ph[6]);
```

```
        Energ_max=Max(dat,5);
        R=-(ph[6]-ph[4]+ph[2]-ph[7])*100./fabs(ph[4]-ph[2]);
        setfillstyle(2,RED);top=10*uy-maxy*dat[0]/Energ_max;
        bar(3*ux,top,4*ux,10*uy);outtextxy(3*ux,top-uy,"|Q(1-2)|");/*Q12*/
        setfillstyle(6,RED);top=10*uy-maxy*dat[1]/Energ_max;
        bar(4*ux,top,5*ux,10*uy);outtextxy(4*ux,top-uy,"|Q(2-3)|");/*Q23*/
        setfillstyle(4,RED);top=10*uy-maxy*dat[2]/Energ_max;
        bar(5*ux,top,6*ux,10*uy);outtextxy(5*ux,top-uy,"|W(5-1)|");/*W51*/
        setfillstyle(1,LIGHTBLUE);top=10*uy-maxy*dat[3]/Energ_max;
        bar(8*ux,top,9*ux,10*uy);outtextxy(8*ux,top-uy,"|W(3-4)|");/*W34*/
        setfillstyle(9,GREEN);top=10*uy-maxy*dat[4]/Energ_max;
        bar(11*ux,top,12*ux,10*uy);outtextxy(11*ux,top-uy,"|Q(4-5)|");/*Q45*/
        settextstyle(SMALL_FONT,HORIZ_DIR,5);
        outtextxy(3*ux,10.5*uy,"SPENT ENERGY");
        outtextxy(8*ux,10.5*uy,"GIVEN ENERGY");
        outtextxy(11*ux,10.5*uy,"LOST ENERGY");
        outtextxy(6*ux,1.5*uy,"          W(3-4)+W(5-1)");
        outtextxy(6*ux,2*uy,"Efficiency = - ---------- =");
        outtextxy(6*ux,2.5*uy,"          |Q(1-2)+Q(2-3)|");
        sprintf(read,"%-5.2f %",R); outtextxy(13.5*ux,2*uy,read);
        break;
case 3: dat[0]=fabs(ph[3]-ph[1]);                    dat[1]=fabs(ph[4]-ph[3]);
        dat[2]=fabs(ph[5]-ph[4]);                    dat[3]=fabs(ph[1]-ph[8]);
        dat[4]=fabs(ph[6]-ph[5]);                    dat[5]=fabs(ph[8]-ph[6]);
        dat[6]=fabs(ph[3]-ph[2]);                    dat[7]=dat[1];
        dat[8]=dat[2];                               dat[9]=fabs(ph[2]-ph[8]);
        dat[10]=fabs(ph[7]-ph[5]);                   dat[11]=fabs(ph[8]-ph[7]);
        Energ_max=Max(dat,12);
        Rs=-(ph[6]-ph[5]+ph[1]-ph[8])*100./fabs(ph[5]-ph[1]);
        Ra=-(ph[7]-ph[5]+ph[2]-ph[8])*100./fabs(ph[5]-ph[2]);
        setfillstyle(2,RED);top=10*uy-maxy*dat[0]/Energ_max;
        bar(1*ux,top,2*ux,10*uy);outtextxy(1*ux,top-uy,"|Q1-2|");
        setfillstyle(4,RED);top=10*uy-maxy*dat[6]/Energ_max;
        bar(2*ux,top,3*ux,10*uy);outtextxy(2*ux,top-uy,"|Q1-2|");
        setfillstyle(2,RED);top=10*uy-maxy*dat[1]/Energ_max;
        bar(3*ux,top,4*ux,10*uy);outtextxy(3*ux,top-uy,"|Q2-3|");
        setfillstyle(4,RED);top=10*uy-maxy*dat[7]/Energ_max;
        bar(4*ux,top,5*ux,10*uy);outtextxy(4*ux,top-uy,"|Q2-3|");
        setfillstyle(2,RED);top=10*uy-maxy*dat[2]/Energ_max;
        bar(5*ux,top,6*ux,10*uy);outtextxy(5*ux,top-uy,"|Q3-4|");
        setfillstyle(4,RED);top=10*uy-maxy*dat[8]/Energ_max;
        bar(6*ux,top,7*ux,10*uy);outtextxy(6*ux,top-uy,"|Q3-4|");
        setfillstyle(2,RED);top=10*uy-maxy*dat[3]/Energ_max;
        bar(7*ux,top,8*ux,10*uy);outtextxy(7*ux,top-uy,"|W6-1|");
        setfillstyle(4,RED);top=10*uy-maxy*dat[9]/Energ_max;
        bar(8*ux,top,9*ux,10*uy);outtextxy(8*ux,top-uy,"|W6-1|");
        setfillstyle(2,LIGHTBLUE);top=10*uy-maxy*dat[4]/Energ_max;
        bar(10*ux,top,11*ux,10*uy);outtextxy(10*ux,top-uy,"|W4-5|");
        setfillstyle(4,LIGHTBLUE);top=10*uy-maxy*dat[10]/Energ_max;
        bar(11*ux,top,12*ux,10*uy);outtextxy(11*ux,top-uy,"|W4-5|");
        setfillstyle(2,GREEN);top=10*uy-maxy*dat[5]/Energ_max;
        bar(13*ux,top,14*ux,10*uy);outtextxy(13*ux,top-uy,"|Q5-6|");
        setfillstyle(4,GREEN);top=10*uy-maxy*dat[11]/Energ_max;
        bar(14*ux,top,15*ux,10*uy);outtextxy(14*ux,top-uy,"|Q5-6|");
        settextstyle(SMALL_FONT,HORIZ_DIR,5);
        outtextxy(ux,10.5*uy,"SPENT ENERGY");
        outtextxy(9.5*ux,10.5*uy,"GIVEN ENERGY");
        outtextxy(13*ux,10.5*uy,"LOST");
        break;
case 4: dat[0]=fabs(ph[3]-ph[1]);dat[1]=fabs(ph[4]-ph[3]);
        dat[2]=fabs(ph[5]-ph[4]);dat[3]=fabs(ph[8]-ph[6]);
        dat[4]=fabs(ph[1]-ph[11]);   dat[5]=fabs(ph[6]-ph[5]);
        dat[6]=fabs(ph[9]-ph[8]);dat[7]=fabs(ph[11]-ph[9]);
        dat[8]=fabs(ph[3]-ph[2]);dat[9]=dat[1];
        dat[10]=dat[2];                  dat[11]=fabs(ph[8]-ph[7]);
        dat[12]=fabs(ph[2]-ph[11]);  dat[13]=fabs(ph[7]-ph[5]);
        dat[14]=fabs(ph[10]-ph[8]);  dat[15]=fabs(ph[11]-ph[10]);
        Energ_max=Max(dat,16);
        Rs=-(ph[6]+ph[9]+ph[1]-h[5]-h[8]-h[11])*100./fabs(ph[5]+ph[8]-ph[1]-ph[6]);
        Ra=-(ph[7]+ph[10]+ph[2]-ph[5]-ph[8]-ph[11])*100./fabs(ph[5]+ph[8]-ph[2]-ph[7]);
        setfillstyle(2,RED);top=10*uy-maxy*dat[0]/Energ_max;
        bar(2*ux,top,2.5*ux,10*uy);outtextxy(2*ux,10.5*uy,"|Q1-2|");
        setfillstyle(4,RED);top=10*uy-maxy*dat[8]/Energ_max;
        bar(2.5*ux,top,3*ux,10*uy);
        setfillstyle(2,RED);top=10*uy-maxy*dat[1]/Energ_max;
        bar(3*ux,top,3.5*ux,10*uy);outtextxy(3*ux,10.5*uy,"|Q2-3|");
        setfillstyle(4,RED);top=10*uy-maxy*dat[9]/Energ_max;
        bar(3.5*ux,top,4*ux,10*uy);
        setfillstyle(2,RED);top=10*uy-maxy*dat[2]/Energ_max;
        bar(4*ux,top,4.5*ux,10*uy);outtextxy(4*ux,10.5*uy,"|Q3-4|");
        setfillstyle(4,RED);top=10*uy-maxy*dat[10]/Energ_max;
        bar(4.5*ux,top,5*ux,10*uy);
        setfillstyle(2,RED);top=10*uy-maxy*dat[3]/Energ_max;
        bar(5*ux,top,5.5*ux,10*uy);outtextxy(5*ux,10.5*uy,"|Q5-5'|");
        setfillstyle(4,RED);top=10*uy-maxy*dat[11]/Energ_max;
        bar(5.5*ux,top,6*ux,10*uy);
        setfillstyle(2,RED);top=10*uy-maxy*dat[4]/Energ_max;
        bar(6*ux,top,6.5*ux,10*uy);outtextxy(6*ux,10.5*uy,"|W6-1|");
        setfillstyle(4,RED);top=10*uy-maxy*dat[12]/Energ_max;
        bar(6.5*ux,top,7*ux,10*uy);
        setfillstyle(2,LIGHTBLUE);top=10*uy-maxy*dat[5]/Energ_max;
        bar(9*ux,top,9.5*ux,10*uy);outtextxy(9*ux,10.5*uy,"|W4-5|");
```

```
            setfillstyle(4,LIGHTBLUE);top=10*uy-maxy*dat[13]/Energ_max;
            bar(9.5*ux,top,10*ux,10*uy);
            setfillstyle(2,LIGHTBLUE);top=10*uy-maxy*dat[6]/Energ_max;
            bar(10*ux,top,10.5*ux,10*uy);outtextxy(10*ux,10.5*uy,"W5'-5''I");
            setfillstyle(4,LIGHTBLUE);top=10*uy-maxy*dat[14]/Energ_max;
            bar(10.5*ux,top,11*ux,10*uy);
            setfillstyle(2,GREEN);top=10*uy-maxy*dat[7]/Energ_max;
            bar(13*ux,top,13.5*ux,10*uy);outtextxy(13*ux,10.5*uy,"IQ5''-6I");
            setfillstyle(4,GREEN);top=10*uy-maxy*dat[15]/Energ_max;
            bar(13.5*ux,top,14*ux,10*uy);
            settextstyle(SMALL_FONT,VERT_DIR,5);
            outtextxy(ux,4.5*uy,"SPENTENERGY");
            outtextxy(8*ux,4.5*uy,"GIVENENERGY");
            outtextxy(12*ux,4.5*uy,"LOST");
            settextstyle(SMALL_FONT,HORIZ_DIR,5);
    default: break;
    }
switch(noc)
    {
    case 1:
    case 2: a=18.*R/5.;
            setfillstyle(8,YELLOW);pieslice(4*ux,2.2*uy,a0,a,1.5*ux);
            setfillstyle(1,BLUE);pieslice(4*ux,2.2*uy,a,360,1.5*ux);
            break;
    case 3:
    case 4: as=18.*Rs/5.;
            ar=18.*Ra/5.;
            setfillstyle(2,YELLOW);pieslice(4*ux,2.2*uy,a0,as,1.5*ux);
            setfillstyle(1,BLUE);pieslice(4*ux,2.2*uy,as,360,1.5*ux);
            setfillstyle(4,YELLOW);pieslice(8*ux,2.2*uy,a0,ar,1.5*ux);
            setfillstyle(1,BLUE);pieslice(8*ux,2.2*uy,ar,360,1.5*ux);
            sprintf(read,"Efficiency =%-5.2f %",Rs); outtextxy(ux,4*uy,read);
            sprintf(read,"Efficiency =%-5.2f %",Ra); outtextxy(7*ux,4*uy,read);
            setfillstyle(2,BROWN);bar(11*ux,uy,12*ux,1.5*uy);
            outtextxy(13*ux,uy,"IDEAL");
            setfillstyle(4,BROWN);bar(11*ux,2*uy,12*ux,2.5*uy);
            outtextxy(13*ux,2*uy,"REAL");
            break;
    default: break;
    }
getch(); closegraph();
}
/*=========================================*/
int Max(double *pdat,int numdat)
{
int i; double val;

val=*pdat;
for(i=0;i at-1;i++)
    if(*(pdat+i+1)val)
        val=*(pdat+i+1);
return(ceil(val));
}
```

RANKRS.C

```
#include "rank.h"

extern int error_signal;
extern double x[12],T[12],P[12],v[12],s[12],h[12];
extern double Ta,Tb,Rp,Rt,Tsc,Ph;


void rankrs()
{
char tecla;

do  {
    dat_rank(2);
    calc(2,Ta,Tb,Rt,Rp,Tsc,Ph);
    if(error_signal==1)
        {
        error_message();
        dat_rank(2);
        calc(2,Ta,Tb,Rt,Rp,Tsc,Ph);}
        do  {
            menu_rank(2);
            tecla=toupper(getch());
            switch(tecla)
                {
                case 'D': clrscr(); res_num_rs(); break;
                case 'G': clrscr(); graf(2); break;
                case 'E': clrscr(); exp_rs(); break;
                case 'M': clrscr(); break;
```

```
                default: printf("\a");
                }
            }
            while(tecla !='M');
        }
    }
    while(tecla !='M');
}
/*=========================================*/
void menu_rank(int noc)
{
textmode(C80); clrscr(); textattr(14+9);
gotoxy(21,2);
switch(noc)
    {
    case 1:cputs("********** IDEAL and SIMPLE **********"); break;
    case 2:cputs("********** REAL and SIMPLE **********"); break;
    case 3:cputs("********** SUPERHEATING **********"); break;
    case 4:cputs("********** REHEATING **********"); break;
    default: break;
    }
gotoxy(21,3); cputs("          ==========          ");
textattr(0+2); gotoxy(21,6); cputs("E === Explanation");
gotoxy(21,8); cputs("D === Datas and statistics");
gotoxy(21,10); cputs("G === Graphic");
gotoxy(21,14); cputs("M === go to Main Menu");
gotoxy(30,23); cputs("Type your choose option");
}
/*=========================================*/
void dat_rank(int noc)
{
int hit;

do  {
    textmode(C80); clrscr(); textattr(14+0);
    gotoxy(21,2); hit=0;
    switch(noc)
        {
        case 1:  cputs("INPUT DATAS  ((Ideal and Simple case))"); break;
        case 2:  cputs("INPUT DATAS  ((Real and Simple case))"); break;
        case 3:  cputs("INPUT DATAS    ((Superheated case))"); break;
        case 4:  cputs("INPUT DATAS      ((Reheated case))"); break;
        default: break;
        }
```



```
textattr(15+3);
switch(noc)
    {
    case 4: gotoxy(1,11);
        cputs("Ph= reheated steam pressure (500<Ph<5000 kPa) (ref.:1000 kPa)");
        gotoxy(1,10); cputs("Tsh=superheater temperature (300<Tsh<600 ºC)(ref.:400ºC)");
        gotoxy(1,6);
        cputs("Ta= water saturation temperature into the boiler (80<Ta<275 ºC)(ref.:250ºC)");
        gotoxy(1,7);
        cputs("Tb=condensation temperature into the condenser (0<Tb<Ta) (ref.:20ºC)");
        gotoxy(1,8); cputs("Et= turbine efficiency (ref.: 70 %) ");
        gotoxy(1,9); cputs("Ep= pump efficiency (ref.: 60 %)");
        break;
    case 3:gotoxy(1,10); cputs("Tsh=superheater temperature (300<Tsh<850ºC)(ref.:400ºC)");
        gotoxy(1,6);
        cputs("Ta= water saturation temperature into the boiler (80<Ta<275ºC)(ref.:250ºC)");
        gotoxy(1,7);
        cputs("Tb=condensation temperature into the condenser (0<Tb<Ta) (ref.:20ºC)");
        gotoxy(1,8); cputs("Et= turbine efficiency (ref.: 70 %)");
        gotoxy(1,9); cputs("Ep= pump efficiency (ref.: 60 %)");
        break;
    case 2:gotoxy(1,8); cputs("Et= turbine efficiency (ref.: 70 %)");
        gotoxy(1,9); cputs("Ep= pump efficiency (ref.: 60 %) ");
    case 1:gotoxy(1,6);
        cputs("Ta= water saturation temperature into the boiler(0<Ta<284ºC)(ref:100ºC)");
        gotoxy(1,7);
        cputs("Tb=condensation temperature into the condenser (0<Tb<Ta)(ref.:20ºC)");
        break;
    default:break;
    }
switch(noc)
    {
    case 4:  gotoxy(30,23); cputs("Ph= ");
    case 3:  gotoxy(30,21); cputs("Tsh=");
    case 2:  gotoxy(30,17); cputs("Et= ");
             gotoxy(30,19); cputs("Ep= ");
    case 1:  gotoxy(30,13); cputs("Ta= ");
             gotoxy(30,15); cputs("Tb= "); break;
    default: break;
    }
switch(noc)
    {
    case 1:  gotoxy(34,13); scanf("%lf",&Ta);
             gotoxy(34,15); scanf("%lf",&Tb);
```

```
                if(Ta284.|| TbTa)
                        { puts("\a"); hit=1; }
                break;
        case 2: gotoxy(34,13); scanf("%lf",&Ta);
                gotoxy(34,15); scanf("%lf",&Tb);
                gotoxy(34,17); scanf("%lf",&Rt);
                gotoxy(34,19); scanf("%lf",&Rp);
                if(Ta284. || TbTa)
                        { puts("\a"); hit=1; }
                break;
        case 3: gotoxy(34,13); scanf("%lf",&Ta);
                gotoxy(34,15); scanf("%lf",&Tb);
                gotoxy(34,17); scanf("%lf",&Rt);
                gotoxy(34,19); scanf("%lf",&Rp);
                gotoxy(34,21); scanf("%lf",&Tsc);
                if(Tsc==600.)   Tsc=601.;
                if(Tsc==800.)   Tsc=801.;
                if(Ta275.||TaP||TbTa||Tsc,||Tsc850.)
                        { puts("\a"); hit=1; }
                break;
        case 4: gotoxy(34,13); scanf("%lf",&Ta);
                gotoxy(34,15); scanf("%lf",&Tb);
                gotoxy(34,17); scanf("%lf",&Rt);
                gotoxy(34,19); scanf("%lf",&Rp);
                gotoxy(34,21); scanf("%lf",&Tsc);
                gotoxy(34,23); scanf("%lf",&Ph);
                if(Tsc==600.)   Tsc=599.;
                if(Ta275.||Ta2||TbTa||Tsc,||Tsc600.||Ph||Ph5000.)
                        { puts("\a"); hit=1; }
                break;
        default: break;
        }
    }
    while(hit==1);
}
/*=========================================*/
void res_num_rs()
{
textmode(C80); clrscr(); textattr(14+0);
gotoxy(5,2); cputs("STATES:");
gotoxy(5,3); cputs("=======");
gotoxy(10,5); cputs("x"); gotoxy(15,5); cputs("T ('C)");
gotoxy(28,5); cputs("P (kPa)"); gotoxy(41,5); cputs("v (m3/kg)");
gotoxy(54,5); cputs("s (kJ/(kgK))"); gotoxy(67,5); cputs("h (kJ/kg)");
gotoxy(10,6); cputs("-   ------        -------        --------- ----------");
textattr(15+0);
gotoxy(5,9); cprintf("1s  %-9.3f      %-6.4f      %-9.4f",P[1],s[1],h[1]);
gotoxy(5,11);cprintf("1a %-9.3f      %-9.4f",P[2],h[2]);
gotoxy(5,13);cprintf("2  %-3.1f      %-6.2f      %-9.3f      %-10.6f%-6.4f%-9.4f",
x[3],T[3],P[3],v[3],s[3],h[3]);
gotoxy(5,15); cprintf("3  %-3.1f      %-6.2f      %-9.3f      %-10.6f%-6.4f%-9.4f",
x[4],T[4],P[4],v[4],s[4],h[4]);
gotoxy(5,17); cprintf("4s %-4.2f      %-6.2f      %-9.3f      %-10.6f%-6.4f%-9.4f",
x[5],T[5],P[5],v[5],s[5],h[5]);
gotoxy(5,19); cprintf("4a %-6.2f      %-9.3f      %-9.4f",T[6],P[6],h[6]);
gotoxy(5,21);cprintf("5  %-3.1f      %-6.2f      %-9.3f      %-10.6f%-6.4f%-9.4f",
x[7],T[7],P[7],v[7],s[7],h[7]);
getch(); clrscr(); textattr(14+0);
gotoxy(25,5); cputs("      W(3-4a) + W(5-1a)");
gotoxy(25,6); cputs("fs,a = - ----------");
gotoxy(25,7); cputs("          3Q(1a-3)3 ");
gotoxy(20,12); cputs("W(3-4a)=h[4a]-h[3]=");
textattr(15+0); cprintf("%f kJ/kg",h[6]-h[4]);
gotoxy(20,14); textattr(14+0); cputs("W(5-1a)=h[1a]-h[5]=");
textattr(15+0); cprintf("%f kJ/kg",h[2]-h[7]);
gotoxy(20,16); textattr(14+0); cputs("Q(1a-2)=h[2]-h[1a]=");
textattr(15+0); cprintf("%f kJ/kg",h[3]-h[2]);
gotoxy(20,18); textattr(14+0); cputs("Q(2-3)=h[3]-h[2]=");
textattr(15+0); cprintf("%f kJ/kg",h[4]-h[3]);
textattr(14+0); gotoxy(20,21); cputs(" ==================");
gotoxy(20,22); cputs("EFFICIENCY : fs,a = ");
textattr(15+0);cprintf("%-10.4f%",-(h[6]-h[4]+h[2]-h[7])*100./fabs(h[4]-h[2]));
textattr(14+0); gotoxy(20,23); cputs(" ==================");
getch(); textmode(C80); clrscr();
graf_resultados(h,2);
}
/*=========================================*/
void ranksup()
{
char tecla;

do  {
    dat_rank(3);
    calc(3,Ta,Tb,Rt,Rp,Tsc,Ph);
    if(error_signal==1)
        {
        error_message();
        dat_rank(3);
```

```
        calc(3,Ta,Tb,Rt,Rp,Tsc,Ph);}
        do  {
            menu_rank(3);
            tecla=toupper(getch());
            switch(tecla)
                {
                case 'D':   clrscr(); res_num_sup(); break;
                case 'G':   clrscr(); graf(3); break;
                case 'E':   clrscr(); exp_sup();
                case 'M':   clrscr(); break;
                default: printf("\a");
                }
            }
            while(tecla !='M');
        }
    }
    while(tecla !='M');
}
/*=============================================*/
void res_num_sup()
{
textmode(C80); clrscr(); textattr(14+0);
gotoxy(5,2); cputs("STATES:");
gotoxy(5,3); cputs("=======");
gotoxy(10,5); cputs("x"); gotoxy(15,5); cputs("T ('C)");
gotoxy(28,5); cputs("P (kPa)"); gotoxy(41,5); cputs("v (m3/kg)");
gotoxy(54,5); cputs("s (kJ/(kgK))"); gotoxy(67,5); cputs("h (kJ/kg)");
gotoxy(10,6); cputs("- ------ ------- --------- ---------- ---------");
textattr(15+0);
gotoxy(5,8); cprintf("1s  %-9.3f      %-6.4f      %-9.4f", P[1],s[1],h[1]);
gotoxy(5,10); cprintf("1a %-9.3f      %-9.4f",P[2],h[2]);
gotoxy(5,12); cprintf("2  %-3.1f      %-6.2f      %-9.3f      %-10.6f%-6.4f%-9.4f",
x[3],T[3],P[3],v[3],s[3],h[3]);
gotoxy(5,14); cprintf("3  %-3.1f      %-6.2f      %-9.3f      %-10.6f%-6.4f%-9.4f",
x[4],T[4],P[4],v[4],s[4],h[4]);
gotoxy(5,16); cprintf("4  %-6.2f      %-9.3f      %-6.4f      %-9.4f",T[5],P[5],s[5],h[5]);
gotoxy(5,18); cprintf("5s %-3.1f      %-6.2f      %-9.3f      %-10.6f%-6.4f%-9.4f",
x[6],T[6],P[6],v[6],s[6],h[6]);
gotoxy(5,20); cprintf("5a %-6.2f      %-9.3f      %-9.4f",T[7],P[7],h[7]);
gotoxy(5,22); cprintf("6  %-3.1f      %-6.2f      %-9.3f      %-10.6f%-6.4f%-9.4f",
x[8],T[8],P[8],v[8],s[8],h[8]);
getch(); clrscr(); textattr(14+0);
gotoxy(25,5); cputs("      W(4-5) + W(6-1)");
gotoxy(25,6); cputs("fsup = - ------------------");
gotoxy(25,7); cputs("      3Q(1-3)+Q(3-4)3 ");
gotoxy(1,11); cputs("W(4-5s)=h[5s]-h[4]=");
textattr(15+0); cprintf("%f kJ/kg",h[6]-h[5]);
gotoxy(1,13); textattr(14+0); cputs("W(6-1s)=h[1s]-h[6]=");
textattr(15+0); cprintf("%f kJ/kg",h[1]-h[8]);
gotoxy(1,15); textattr(14+0); cputs("Q(1s-2)=h[2]-h[1s]=");
textattr(15+0); cprintf("%f kJ/kg",h[3]-h[1]);
gotoxy(1,17); textattr(14+0); cputs("Q(2-3)=h[3]-h[2]=");
textattr(15+0); cprintf("%f kJ/kg",h[4]-h[3]);
gotoxy(1,19); textattr(14+0); cputs("Q(3-4)=h[4]-h[3]=");
textattr(15+0); cprintf("%f kJ/kg",h[5]-h[4]);
gotoxy(42,11); textattr(14+0);c puts("W(4-5a)=h[5a]-h[4]=");
textattr(15+0); cprintf("%f kJ/kg",h[7]-h[5]);
gotoxy(42,13); textattr(14+0); cputs("W(6-1a)=h[1a]-h[6]=");
textattr(15+0); cprintf("%f kJ/kg",h[2]-h[8]);
gotoxy(42,15); textattr(14+0); cputs("Q(1a-2)=h[2]-h[1a]=");
textattr(15+0); cprintf("%f kJ/kg",h[3]-h[2]);
gotoxy(42,17); textattr(14+0); cputs("Q(2-3)=h[3]-h[2]=");
textattr(15+0); cprintf("%f kJ/kg",h[4]-h[3]);
gotoxy(42,19); textattr(14+0); cputs("Q(3-4)=h[4]-h[3]=");
textattr(15+0); cprintf("%f kJ/kg",h[5]-h[4]);
textattr(14+0); gotoxy(1,21);
cputs("==========                    ============");
gotoxy(1,22); cputs("EFFICIENCY : fsup,s = ");
textattr(15+0);cprintf("%-10.4f%",-(h[6]-h[5]+h[1]-h[8])*100./fabs(h[5]-h[1]));
gotoxy(42,22); textattr(14+0); cputs(" fsup,a = ");
textattr(15+0);cprintf("%-10.4f%",-(h[7]-h[5]+h[2]-h[8])*100./fabs(h[5]-h[2]));
textattr(14+0); gotoxy(1,23);
cputs("==========          ============");
getch(); textmode(C80); clrscr(); graf_resultados(h,3);
}
/*=============================================*/
void rankrec()
{
char tecla;

do  {
    dat_rank(4);
    calc(4,Ta,Tb,Rt,Rp,Tsc,Ph);
    if(error_signal==1)
        {
        error_message();
        dat_rank(4);
        calc(4,Ta,Tb,Rt,Rp,Tsc,Ph);}
```

```
        do {
            menu_rank(4);
            tecla=toupper(getch());
            switch(tecla)
                {
                case 'D': clrscr(); res_num_rec(); break;
                case 'G': clrscr(); graf(4); break;
                case 'E': clrscr(); exp_rec();
                case 'M': clrscr(); break;
                default: printf("\a");
                }
            }
            while(tecla !='M');
        }
    }while(tecla !='M');
}
/*=========================================*/
void res_num_rec()
{
textmode(C80); clrscr(); textattr(14+0);
gotoxy(3,2); cputs("STATES:");
gotoxy(3,3); cputs("=======");
gotoxy(10,5); cputs("x"); gotoxy(15,5); cputs("T ('C)");
gotoxy(28,5); cputs("P (kPa)"); gotoxy(41,5); cputs("v (m3/kg)");
gotoxy(54,5); cputs("s (kJ/(kgK))"); gotoxy(67,5); cputs("h (kJ/kg)");
gotoxy(10,6);cputs("- ------ ------- --------- ---------- ---------");
textattr(15+0);
gotoxy(3,8); cprintf("1s: %-9.3f %-6.4f %-9.4f",P[1],s[1],h[1]);
gotoxy(3,9); cprintf("1a: %-9.3f %-9.4f",P[2],h[2]);
gotoxy(3,11); cprintf("2: %-3.1f %-6.2f %-9.3f %-10.6f%-6.4f%-9.4f",
x[3],T[3],P[3],v[3],s[3],h[3]);
gotoxy(3,13);cprintf("3: %-3.1f %-6.2f %-9.3f %-10.6f%-6.4f%-9.4f",
x[4],T[4],P[4],v[4],s[4],h[4]);
gotoxy(3,15); cprintf("4: %-6.2f %-9.3f %-6.4f %-9.4f",T[5],P[5],s[5],h[5]);
gotoxy(3,17);cprintf("5s: %-6.2f %-9.3f %-6.4f %-9.4f",T[6],P[6],s[6],h[6]);
gotoxy(3,18);cprintf("5a: %-9.3f %-9.4f",P[7],h[7]);
gotoxy(3,20);cprintf("5': %-6.2f %-9.3f %-6.4f %-9.4f",T[8],P[8],s[8],h[8]);
gotoxy(3,22);cprintf("5''s: %-3.1f %-6.2f %-9.3f %-10.6f%-6.4f%-9.4f",
x[9],T[9],P[9],v[9],s[9],h[9]);
gotoxy(3,23); cprintf("5''a: %-6.2f %9.3f %-9.4f",T[10],P[10],h[10]);
gotoxy(3,25);cprintf("6: %-3.1f %-6.2f %-9.3f %-10.6f%-6.4f%-9.4f",
x[11],T[11],P[11],v[11],s[11],h[11]);
getch(); clrscr(); textattr(14+0);
gotoxy(20,2); cputs(" W(4-5) + W(5'-5'') + W(6-1)");
gotoxy(20,3); cputs("frec = - ----------");
gotoxy(20,4); cputs(" 3Q(1-3)+Q(3-4)+Q(5-5')3 ");
gotoxy(10,6); textattr(15+0); cputs("(Todas las magnitudes de trabajo y calor estn en kJ/kg)");
gotoxy(1,7); textattr(14+0); cputs("W(4-5s)=h[5s]-h[4]=");
textattr(15+0); cprintf("%f",h[6]-h[5]);
gotoxy(1,9); textattr(14+0); cputs("W(5'-5''s)=h[5''s]-h[5']=");
textattr(15+0); cprintf("%f",h[9]-h[8]);
gotoxy(1,11); textattr(14+0); cputs("W(6-1s)=h[1s]-h[6]=");
textattr(15+0); cprintf("%f",h[1]-h[11]);
gotoxy(1,13); textattr(14+0); cputs("Q(1s-2)=h[2]-h[1s]=");
textattr(15+0); cprintf("%f",h[3]-h[1]);
gotoxy(1,15); textattr(14+0); cputs("Q(2-3)=h[3]-h[2]=");
textattr(15+0); cprintf("%f",h[4]-h[3]);
gotoxy(1,17); textattr(14+0); cputs("Q(3-4)=h[4]-h[3]=");
textattr(15+0); cprintf("%f",h[5]-h[4]);
gotoxy(1,19); textattr(14+0); cputs("Q(5s-5')=h[5']-h[5s]=");
textattr(15+0); cprintf("%f",h[8]-h[6]);
gotoxy(42,7); textattr(14+0); cputs("W(4-5a)=h[5a]-h[4]=");
textattr(15+0); cprintf("%f",h[7]-h[5]);
gotoxy(42,9); textattr(14+0); cputs("W(5'-5''a)=h[5''a]-h[5']=");
textattr(15+0); cprintf("%f",h[10]-h[8]);
gotoxy(42,11); textattr(14+0); cputs("W(6-1a)=h[1a]-h[6]=");
textattr(15+0); cprintf("%f",h[2]-h[11]);
gotoxy(42,13); textattr(14+0); cputs("Q(1a-2)=h[2]-h[1a]=");
textattr(15+0); cprintf("%f",h[3]-h[2]);
gotoxy(42,15); textattr(14+0); cputs("Q(2-3)=h[3]-h[2]=");
textattr(15+0); cprintf("%f",h[4]-h[3]);
gotoxy(42,17); textattr(14+0); cputs("Q(3-4)=h[4]-h[3]=");
textattr(15+0);cprintf("%f",h[5]-h[4]);
gotoxy(42,19); textattr(14+0); cputs("Q(5a-5')=h[5']-h[5a]=");
textattr(15+0); cprintf("%f",h[8]-h[7]);
textattr(14+0); gotoxy(1,21); cputs(" ======= ========");
gotoxy(1,22); cputs("EFFICIENCY : frec,s = ");
textattr(15+0);cprintf("%-10.4f%",-(h[6]+h[9]+h[1]-h[5]-h[8]-h[11])*100./
fabs(h[5]+h[8]-h[1]-h[6]));
gotoxy(42,22); textattr(14+0); cputs(" frec,a = ");
textattr(15+0);cprintf("%-10.4f%",-(h[7]+h[10]+h[2]-h[5]-h[8]-h[11])*100./
fabs(h[6]+h[8]-h[2]-h[7]));
textattr(14+0);gotoxy(1,23);
cputs(" ======== ========");
getch(); textmode(C80); clrscr();
graf_resultados(h,4);
}
```

RANKCALC.C

```
#include "rank.h"

extern double x[12],T[12],P[12],v[12],s[12],h[12];
extern int error_signal;

double Pls[76]={
    .61,.87,1.23,1.71,2.34,3.17,4.24,5.62,7.37,9.58,12.33,15.74,
    19.92,25.,31.15,38.54,47.35,57.80,70.1,84.52,101.32,120.80,143.27,
    169.07,198.55,232.11,270.15,313.09,361.39,415.53,475.99,543.30,618.,
    700.66,791.86,892.2,1002.3,1122.9,1254.5,1398.,1553.9,1723.1,1906.3,
    2104.3,2317.8,2547.8,2795.,3060.3,3344.7,3649.,3974.2,4321.3,4691.2,
    5085.,5503.8,5948.6,6420.5,6920.8,7450.6,8011.1,8603.7,9214.4,9869.4,
    10561,11289,12056,12862,13712,14605,15545,16535,17577,18675,19833,21054,
    22050
    };
double hvsa[76]={
    2500.018,2510.635,2520.414,2529.744,2538.851,2547.853,2556.81,
    2565.741,2574.646,2583.518,2592.343,2601.109,2609.803,2618.416,2626.939,
    2635.365,2643.689,2651.904,2660.005,2667.986,2675.841,2683.562,
    2691.141,2698.57,2705.839,2712.938,2719.855,2726.579,2733.099,2739.401,
    2745.474,2751.304,2756.88,2762.188,2767.216,2771.953,2776.387,2780.506,
    2784.3,2787.758,2790.871,2793.628,2796.022,2798.043,2799.684,2800.937,
    2801.796,2802.253,2802.303,2801.94,2801,2799.3952,2798.318,2796.252,
    2793.748,2786,2784,2777,2765,2758,2744,2737,2724,2708,2690,2675,
    2650,2620,2590,2550,2500,2441,2380,2295,2210,2099
    }; /*ampliada*/
double hls[76]={
    .0,21.0,42.0,63.0,83.9,104.8,125.6,146.5,167.4,188.3,209.1,
    230.0,250.9,271.9,292.8,313.7,334.7,355.7,376.7,397.8,418.9,440.0,461.1,
    482.3,503.5,524.8,546.1,567.4,588.8,610.3,631.9,653.5,675.2,697.0,
    718.8,740.8,762.8,785.0,807.2,829.6,852.1,874.7,897.5,920.4,943.5,966.7,
    990.1,1013.7,1037.5,1061.4,1085.6,1110.1,1134.8,1159.8,1185.1,1210.7,
    1236.6,1263.0,1289.8,1317.1,1344.9,1373.3,1402.3,1432.1,1462.6,1494.0,
    1526.3,1559.7,1594.3,1632.5,1671.8,1713.9,1761.6,1817.8,1890.1,2099.3
    };
double rx,rt,rp,rv,rs,rh;
double ins1,ins2,inh1,inh2,inv1,inv2;
/*-------------------------------------------------------------------*/
double Tls[76]={
    0,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,
    105,110,115,120,125,130,135,140,145,150,155,160,165,170,175,180,185,190,
    195,200,205,210,215,220,225,230,235,240,245,250,255,260,265,270,275,
    280,285,290,295,300,305,310,315,320,325,330,335,340,345,350,355,360,365,
    370,374.4
    };
double sls[76]={
    .0012,.0757,.1509,.2244,.2965,.3672,.4367,.5049,.5720,.6381,
    .7031,.7672,.8303,.8926,.9540,1.0146,1.0744,1.1335,1.1917,1.2493,1.3062,
    1.3624,1.4179,1.4728,1.5271,1.5807,1.6338,1.6864,1.7384,1.7899,1.8409,
    1.8915,1.9416,1.9912,2.0405,2.0894,2.1380,2.1862,2.2341,2.2817,2.3290,
    2.3761,2.4230,2.4696,2.5161,2.5623,2.6084,2.6544,2.7002,2.7460,2.7917,
    2.8373,2.8829,2.9286,2.9743,3.0200,3.0660,3.1121,3.1585,3.2052,3.2523,
    3.3000,3.3483,3.3973,3.4473,3.4984,3.5507,3.6045,3.6601,3.7176,3.7775,
    3.8400,3.9056,3.9746,4.0476,4.4298
    };
double vls[76]={
    .001,.001,.001,.001001,.001002,.001003,.001004,.001006,.001008,
    .001010,.001012,.001014,.001017,.001020,.001023,.001026,.001029,
    .001032,.001036,.001039,.001043,.001047,.001051,.001056,.001060,.001065,
    .001070,.001075,.001080,.001085,.001091,.001096,.001102,.001108,
    .001114,.001121,.001127,.001134,.001141,.001148,.001156,.001164,.001172,
    .001180,.001189,.001198,.001208,.001218,.001228,.001239,.001250,
    .001262,.001275,.001288,.001302,.001317,.001333,.001349,.001366,.001385,
    .001404,.001425,.001447,.001470,.001499,.001528,.001561,.001598,
    .001639,.001686,.001741,.001808,.001896,.002016,.002225,.00315
    };
/*-------------------------------------------------------------------*/
double svs[58]={
    9.157787,9.026691,8.901974,8.782665,8.668362,8.559106,8.454624,
    8.354491,8.258406,8.166168,8.077588,7.992487,7.910687,7.832011,
    7.756285,7.683353,7.613065,7.545275,7.47983,7.416585,7.355406,7.296193,
    7.238843,7.183254,7.129329,7.076965,7.026079,6.976594,6.928433,6.881517,
    6.835767,6.79111,6.747472,6.704781,6.662964,6.621951,6.581671,6.542098,
    6.503231,6.465072,6.427618,6.390848,6.354361,6.318122,6.282132,
    6.246392,6.210901,6.175658,6.140619,6.105706,6.070842,6.035946,6.00094,
    5.965749,5.930303,5.894536,5.858385,5.829143
    };
double hvs[58]={
    2500.018,2510.635,2520.414,2529.744,2538.851,2547.853,2556.81,
    2565.741,2574.646,2583.518,2592.343,2601.109,2609.803,2618.416,2626.939,
    2635.365,2643.689,2651.904,2660.005,2667.986,2675.841,2683.562,
    2691.141,2698.57,2705.839,2712.938,2719.855,2726.579,2733.099,2739.401,
    2745.474,2751.304,2756.88,2762.188,2767.216,2771.953,2776.387,2780.506,
    2784.3,2787.758,2790.871,2793.628,2796.022,2798.043,2799.684,2800.937,
    2801.796,2802.253,2802.303,2801.94,2801.158,2799.3952,2798.318,2796.252,
```

```
    2793.748,2790.803,2787.414,2784.379
    };
double vvs[58]={
    206.131180,147.19986,106.360039,78.036652,57.801746,43.447338,
    32.907543,25.251165,19.537264,15.262927,12.046964,9.578062,7.678644,
    6.200582,5.046193,4.133783,3.408463,2.828611,2.361404,1.981656,1.669911,
    1.415218,1.207354,1.036100,.892058,.770447,.668091,.581816,.508473,
    .445708,.392089,.346233,.306759,.27242,.242519,.216497,.193791,
    .173851,.156275,.140790,.12713,.115027,.104245,.09463,.086046,.078355,
    .071428,.065214,.059739,.054772,.050182,.045969,.042134,.038669,.03554,
    .032703,.030117,.028201
    };
/*-------------------------------------------------------------------*/
doubleTv[16]={0,20,40,60,80,100,120,140,160,180,200,220,240,260,280,284};
doublexv[11]={0.,.1,.2,.3,.4,.5,.6,.7,.8,.9,1.};
double hv[16][11]={
    {0.,250.0048,500.0063,750.0077,1000.009,1250.011,1500.012,1750.013,
    2000.015,2250.016,2500.018},
    {83.9,329.3638,574.8623,820.3609,1065.859,1311.358,1556.857,1802.355,
    2047.854,2293.352,2538.851},
    {167.4,408.1039,648.8308,889.5577,1130.285,1371.012,1611.739,1852.465,
    2093.192,2333.919,2574.646},
    {250.9,486.8199,722.7069,958.5938,1194.481,1430.368,1666.255,1902.142,
    2138.029,2373.916,2609.803},
    {334.7,565.614,796.5112,1027.408,1254.306,1489.203,1720.1,1950.997,
    2181.895,2412.792,2643.689},
    {418.9,644.5519,870.2507,1095.949,1321.648,1547.347,1773.046,1998.744,
    2224.443,2450.142,2675.841},
    {503.5,723.7277,943.9622,1164.197,1384.431,1604.666,1824.901,2045.135,
    2265.37,2485.605,2705.839},
    {588.8,803.2711,1017.696,1232.122,1446.547,1660.972,1875.398,2089.823,
    2304.248,2518.674,2733.099},
    {675.2,883.3513,1091.521,1299.691,1507.861,1716.031,1924.2,2132.37,
    2340.54,2548.71,2756.88},
    {762.8,964.1754,1165.532,1366.889,1568.246,1769.603,1970.96,2172.316,
    2373.673,2575.03,2776.387},
    {852.1,1045.991,1239.867,1433.742,1627.618,1821.493,2015.369,2209.244,
    2403.12,2596.995,2790.871},
    {943.5,1129.108,1314.727,1500.347,1685.967,1871.586,2057.206,2242.825,
    2428.445,2614.064,2799.684},
    {1037.5,1213.946,1390.43,1566.914,1743.398,1919.882,2096.367,2272.851,
    2449.335,2625.819,2802.303},
    {1134.8,1301.142,1467.495,1633.848,1800.201,1966.553,2132.907,2299.259,
    2465.612,2631.965,2798.318},
    {1236.6,1391.721,1546.798,1701.875,1856.952,2012.029,2167.106,2322.183,
    2477.26,2632.337,2787.414},
    {1257.707,1410.374,1563.042,1715.709,1868.376,2021.043,2173.71,2326.378,
    2479.045,2631.712,2784.379}    };
double sv[16][11]={
    {.0012,.9146801,1.830581,2.746482,3.662383,4.578283,5.494185,6.410085,
    7.325986,8.241886,9.157787},
    {.2965,1.133708,1.970892,2.808076,3.645259,4.482443,5.319627,6.15681,
    6.993994,7.831178,8.668362},
    {.572,1.340648,2.109288,2.877928,3.646567,4.415207,5.183847,5.952487,
    6.721126,7.489766,8.258406},
    {.8303,1.53835,2.246388,2.954425,3.662462,4.3705,5.078538,5.786574,6.494612,
    7.202649,7.910687},
    {1.0744,1.728278,2.382143,3.036008,3.689874,4.343739,4.997604,5.651469,
    6.305335,6.959199,7.613065},
    {1.3062,1.91111,2.516033,3.120954,3.725876,4.330798,4.93572,5.540641,
    6.145563,6.750485,7.355406},
    {1.5271,2.087299,2.647525,3.207751,3.767976,4.328202,4.888427,5.448652,
    6.008878,6.569103,7.129329},
    {1.7384,2.257408,2.776411,3.295414,3.814416,4.333419,4.852422,5.371425,
    5.890428,6.409431,6.928433},
    {1.9416,2.422156,2.902747,3.383337,3.863928,4.344518,4.825109,5.3057,
    5.78629,6.266881,6.747472},
    {2.138,2.582348,3.026717,3.471086,3.915455,4.359825,4.804194,5.248563,
    5.692933,6.137301,6.581671},
    {2.329,2.738898,3.148756,3.558613,3.968471,4.378329,4.788187,5.198045,
    5.607902,6.01776,6.427618},
    {2.5161,2.892661,3.269269,3.645877,4.022485,4.399093,4.775701,5.152309,
    5.528916,5.905524,6.282132},
    {2.7002,3.04428,3.38832,3.732358,4.076395,4.420432,4.76447,5.108507,
    5.452544,5.796582,6.140619},
    {2.8829,3.194739,3.50654,3.81834,4.130139,4.44194,4.75374,5.06554,5.37734,
    5.68914,6.00094},
    {3.066,3.345197,3.62444,3.903683,4.182927,4.46217,4.741413,5.020656,
    5.299899,5.579142,5.858385},
    {3.102829,3.37546,3.648092,3.920723,4.193355,4.465986,4.738617,5.011249,
    5.28388,5.556511,5.829143}
    };
double vv[16][11]={
    {.001,20.614018,41.227039,61.840057,82.453072,103.060086,123.679108,
    144.292130,164.905151,185.518158,206.13118},
    {.001002,5.781076,11.561152,17.341225,23.1213,28.901373,34.68145,40.461521,
    46.2416,52.021667,57.801746},
    {.001008,1.954633,3.908259,5.861885,7.815511,9.769136,11.722762,13.676388,
```

```
    3809.7,3921.9,4035.8,4151.5,4268.9},
    {0.,2752.2,2860.7,2964.1,3066.2,3168.6,3271.9,3376.3,3482.2,3589.5,3698.5,
    3809.,3921.3,4035.3,4151.,4268.4},
    {0.,0.,2850.2,2957.2,3061.3,3164.8,3268.9,3373.8,3480.1,3587.7,3696.9,
    3807.7,3920.1,4034.2,4150.,4267.6},
    {0.,0.,2839.1,2950.1,3056.2,3161.,3265.8,3371.4,3478.,3586.,3695.4,3806.3,
    3918.9,4033.1,4149.1,4266.7},
    {0.,0.,2827.3,2942.8,3051.1,3157.1,3262.7,3368.8,3475.9,3584.2,3693.8,
    3805.,3917.7,4032.,4148.1,4265.8},
    {0.,0.,0.,2923.4,3037.8,3147.2,3255.,3362.5,3470.6,3579.7,3689.9,3801.5,
    3914.7,4029.4,4145.7,4263.6},
    {0.,0.,0.,2902.4,3023.9,3137.1,3247.1,3356.1,3465.3,3575.1,3686.,3798.1,
    3911.6,4026.7,4143.2,4261.5},
    {0.,0.,0.,2879.7,3009.3,3126.6,3239.1,3349.7,3459.9,3570.6,3682.1,3794.7,
    3908.6,4024.,4140.8,4259.3},
    {0.,0.,0.,2854.9,2994.1,3115.9,3230.9,3343.1,3454.5,3566.,3678.1,3791.3,
    3905.6,4021.2,4138.4,4257.1},
    {0.,0.,0.,0.,2961.2,3093.4,3214.1,3329.8,3443.6,3556.8,3670.2,3784.3,3899.5,
    4015.8,4133.5,4252.7},
    {0.,0.,0.,0.,2924.7,3069.6,3196.6,3316.2,3432.5,3547.5,3662.3,3777.4,3893.4,
    4010.4,4128.7,4248.3},
    {0.,0.,0.,0.,2884.,3044.2,3178.6,3302.3,3421.3,3538.1,3654.2,3770.4,3887.2,
    4004.9,4123.8,4243.9}
    };
double ssh[16][16]={
    {8.4498,8.6893,8.9040,9.0996,9.2799,9.4476,9.6048,9.753,9.8935,10.027,
    10.155,10.278,10.396,10.510,10.620,10.727},
    {7.6959,7.9413,8.1583,8.3551,8.5360,8.7040,8.8614,9.0098,9.1504,9.2843,
    9.4123,9.5351,9.6532,9.7672,9.8775,9.9843},
    {7.3610,7.6146,7.8347,8.0329,8.2146,8.3831,8.5407,8.6893,8.8300,8.9640,
    9.0921,9.2149,9.3331,9.4471,9.5574,9.6643},
    {0.,7.2804,7.5072,7.7084,7.8916,8.0610,8.2192,8.3682,8.5092,8.6434,8.7716,
    8.8945,9.0128,9.1268,9.2372,9.3441},
    {0.,7.0779,7.3122,7.5165,7.7014,7.8717,8.0305,8.1798,8.3211,8.4554,8.5838,
    8.7068,8.8252,8.9393,9.0497,9.1566},
    {0.,6.9287,7.1712,7.3789,7.5655,7.7367,7.8961,8.0458,8.1873,8.3219,8.4504,
    8.5735,8.6919,8.8062,8.9166,9.0236},
    {0.,0.,6.9669,7.1819,7.3719,7.5451,7.7057,7.8562,7.9982,8.1332,8.2619,8.3853,
    8.5039,8.6182,8.7287,8.8358},
    {0.,0.,6.8156,7.0388,7.2326,7.4079,7.5697,7.7209,7.8635,7.9988,8.1278,8.2514,
    8.3702,8.4846,8.5953,8.7024},
    {0.,0.,6.6930,6.9251,7.1229,7.3003,7.4633,7.6154,7.7585,7.8942,8.0235,8.1473,
    8.2662,8.3808,8.4916,8.5988},
    {0.,0.,0.,6.7093,6.9183,7.1014,7.2677,7.4219,7.5664,7.7030,7.8331,7.9574,
```



```
    15.630014,17.583639,19.537264},
    {.001017,.76878,1.536542,2.304305,3.072068,3.839831,4.607594,5.375356,
    6.143119,6.910882,7.678644},
    {.001029,.341772,.682516,1.023259,1.364003,1.704746,2.045489,2.386233,
    2.726976,3.06772,3.408463},
    {.001043,.16793,.334817,.501704,.66859,.835477,1.002364,1.169251,1.336138,
    1.503024,1.669911},
    {.001060,.09016,.17926,.268359,.357459,.446559,.535659,.624758,.713858,
    .802958,.892058},
    {.00108,.051819,.102558,.153298,.204037,.254776,.305516,.356255,.406994,
    .45773,.508473},
    {.001102,.031668,.062233,.092799,.123365,.15393,.184496,.215062,.245627,
    .276193,.306759},
    {.001127,.020394,.03966,.058927,.078193,.097459,.116726,.135992,.155259,
    .174525,.193791},
    {.001156,.013753,.026351,.038948,.051546,.064143,.07674,.089338,.101935,
    .114532,.12713},
    {.001189,.009675,.01816,.026646,.035132,.043617,.052103,.060589,.069074,
    .07756,.086046},
    {.001228,.007079,.01293,.018781,.024633,.030484,.036335,.042186,.048037,
    .0538888,.059739},
    {.001275,.005361,.009447,.013533,.017619,.021704,.02579,.029876,.033962,
    .038048,.042134},
    {.001333,.004211,.007089,.009968,.012846,.015725,.018603,.021482,.02436,
    .027239,.030117},
    {.001346,.004031,.006717,.009402,.012088,.014773,.017459,.020144,.02283,
    .025515,.028201}
    };
/*-------------------------------------------------------------------*/
int a[16]={0,0,0,1,1,1,2,2,2,3,3,3,3,4,4,4};
double Tsh[16]={100.,150.,200.,250.,300.,350.,400.,450.,500.,550.,600.,650.,700.,
        750.,800.,850.};
double Psh[16]={10.,50.,100.,200.,300.,400.,600.,800.,1000.,1500.,2000.,2500.,
        3000.,4000.,5000.,6000.};
double hsh[16][16]={
    {2688.1,2783.3,2879.5,2976.8,3075.6,3175.9,3277.7,3381.1,3486.2,3593.,
    3701.5,3811.7,3923.7,4037.4,4152.9,4270.2},
    {2682.9,2780.5,2877.6,2975.6,3074.7,3175.1,3277.1,3380.6,3485.8,3592.6,
    3701.2,3811.4,3923.4,4037.2,4152.7,4270.},
    {2676.,2776.8,2875.3,2974.,3073.5,3174.2,3276.4,3380.,3485.3,3592.2,3700.8,
    3811.1,3923.1,4036.9,4152.4,4269.8},
    {0.,2769.1,2870.6,2970.7,3071.1,3172.4,3274.9,3378.8,3484.3,3591.3,3700.,
    3810.4,3922.5,4036.4,4152.,4269.3},
    {0.,2760.9,2865.7,2967.4,3068.7,3170.5,3273.4,3377.6,3483.2,3590.4,3699.2,
```

```
    8.0767,8.1917,8.3027,8.4101},
    {0.,0.,0.,6.5451,6.7671,6.9565,7.1263,7.2826,7.4286,7.5662,7.6970,7.8218,
    7.9416,8.0569,8.1681,8.2758},
    {0.,0.,0.,6.4076,6.6446,6.8409,7.0145,7.1730,7.3205,7.4592,7.5906,7.7161,
    7.8362,7.9518,8.0633,8.1712},
    {0.,0.,0.,6.2855,6.5399,6.7437,6.9213,7.0822,7.2312,7.3709,7.5031,7.6291,
    7.7497,7.8656,7.9774,8.0855},
    {0.,0.,0.,0.,6.3622,6.5835,6.7699,6.9358,7.0879,7.2298,7.3636,7.4907,7.6121,
    7.7287,7.8411,7.9496},
    {0.,0.,0.,0.,6.2085,6.4512,6.6474,6.8188,6.9743,7.1184,7.2538,7.3820,7.5044,
    7.6216,7.7345,7.8435},
    {0.,0.,0.,0.,6.0669,6.3354,6.5429,6.7202,6.8793,7.0258,7.1627,7.2922,7.4154,
    7.5333,7.6467,7.7562}
    };
/*=========================================*/
void calc(int noc,double Ta,double Tb,double Rt,double Rp,double Tsc,double Ph)
{
double entrop;

error_signal=0;
switch(noc)
    {
    case 1: T[2]=Ta; T[3]=Ta; T[4]=Tb; T[5]=Tb;          /*Ideal y simple*/
            x[2]=0.; x[3]=1.; x[5]=0.;
            liqsat(Ta); P[2]=rp; v[2]=rv; s[2]=rs; h[2]=rh;
            liqsat(Tb); P[5]=rp; v[5]=rv; s[5]=rs; h[5]=rh;
            P[1]=P[2]; P[3]=P[2]; P[4]=P[5]; s[1]=s[5];
            h[1]=h[5]+v[5]*(P[1]-P[5]);
            vapsat(T[3]); v[3]=rv; s[3]=rs; h[3]=rh;
            s[4]=s[3];
            vapor(T[4],s[4]); x[4]=rx; h[4]=rh; v[4]=rv;
            break;
    case 2: T[3]=Ta; T[4]=Ta; T[5]=Tb; T[6]=Tb; T[7]=Tb;   /*Real y simple*/
            liqsat(T[3]); P[3]=rp; v[3]=rv; s[3]=rs; h[3]=rh;
            liqsat(T[7]); P[7]=rp; v[7]=rv; s[7]=rs; h[7]=rh;
            P[1]=P[3]; P[2]=P[3]; P[4]=P[3]; P[5]=P[7]; P[6]=P[7];
            x[3]=0.; x[7]=0.; x[4]=1; s[1]=s[7];
            vapsat(T[4]); v[4]=rv; s[4]=rs; h[4]=rh;
            s[5]=s[4]; h[1]=h[7]+v[7]*(P[1]-P[7]);
            vapor(T[5],s[5]); x[5]=rx; h[5]=rh; v[5]=rv;
            h[6]=h[4]+Rt*(h[5]-h[4])/100.;
            h[2]=h[7]+100.*(h[1]-h[7])/Rp;
            entrop=vapsat_return_h(T[7]);
            if(h[6]entrop)
                error_signal=1;
            break;
    case 3: T[3]=Ta; T[4]=Ta; T[6]=Tb; T[7]=Tb; T[8]=Tb; /*Supercalentado*/
            x[3]=0.; x[8]=0.; x[4]=1;
            liqsat(T[3]); P[3]=rp; v[3]=rv; s[3]=rs; h[3]=rh;
            liqsat(T[8]); P[8]=rp; v[8]=rv; s[8]=rs; h[8]=rh;
            P[1]=P[3]; P[2]=P[3]; P[4]=P[3]; P[5]=P[3]; P[6]=P[8]; P[7]=P[8];
            s[1]=s[8];h[1]=h[8]+v[8]*(P[1]-P[8]);
            h[2]=h[8]+100.*(h[1]-h[8])/Rp;
            vapsat(T[4]); v[4]=rv; s[4]=rs; h[4]=rh; T[5]=Tsc;
            vapsupercal(T[5],P[5]); h[5]=rh; s[5]=rs; s[6]=s[5];
            vapor(T[6],s[6]); x[6]=rx; h[6]=rh; v[6]=rv;
            h[7]=h[5]+Rt*(h[6]-h[5])/100.;
            entrop=vapsat_return_h(T[8]);
            if(h[7]entrop)
                error_signal=1;
            break;
    case 4: x[11]=0.; x[3]=0.; x[4]=1.;          /*Recalentado*/
            T[3]=Ta; T[4]=Ta; T[9]=Tb; T[10]=Tb; T[11]=Tb;
            T[5]=Tsc;T[8]=Tsc;
            liqsat(T[3]); P[3]=rp; v[3]=rv; s[3]=rs; h[3]=rh;
            liqsat(T[11]); P[11]=rp; v[11]=rv; s[11]=rs; h[11]=rh;
            P[9]=P[11]; P[10]=P[11]; P[1]=P[3]; P[2]=P[3]; P[4]=P[3];
            P[5]=P[3]; P[6]=Ph; P[7]=Ph; P[8]=Ph;
            s[1]=s[11];
            vapsat(T[4]); v[4]=rv; s[4]=rs; h[4]=rh;
            h[1]=h[11]+v[11]*(P[1]-P[11]);
            h[2]=h[11]+100.*(h[1]-h[11])/Rp;
            vapsupercal(T[5],P[5]); h[5]=rh; s[5]=rs; s[6]=s[5];
            vapsupercal2(P[6],s[6]); T[6]=rt; h[6]=rh;
            h[7]=h[5]+Rt*(h[6]-h[5])/100.;
            vapsupercal(T[8],P[8]); h[8]=rh; s[8]=rs; s[9]=s[8];
            vapor(T[9],s[9]); x[9]=rx; h[9]=rh; v[9]=rv;
            h[10]=h[8]+Rt*(h[9]-h[8])/100.;
            entrop=vapsat_return_h(T[11]);
            if(h[10]entrop)
                error_signal=1;
            break;
    default: break;
    }
}
/*=========================================*/
```

```
void liqsat(double Temp)      /* Líquido saturado */
{
int cont, i;

cont=i=0;
while(iL && cont==0)
    {
    if(Temp==Tls[i])
        {
        rp=Pls[i], rv=vls[i], rs=sls[i], rh=hls[i];
        cont=1;
        }
    else if(TempTls[i] && Temp [i+1])
        {
        rp=intrplcn(Tls[i],Tls[i+1],Pls[i],Pls[i+1],Temp);
        rv=intrplcn(Tls[i],Tls[i+1],vls[i],vls[i+1],Temp);
        rs=intrplcn(Tls[i],Tls[i+1],sls[i],sls[i+1],Temp);
        rh=intrplcn(Tls[i],Tls[i+1],hls[i],hls[i+1],Temp);
        cont=1;
        }
    i++;
    }
}
/*=========================================*/
void vapsat(double Temp)              /* Estado vapor saturado */
{
int i, cont;

cont=i=0;
while(i: && cont==0)
    {
    if(Temp==Tls[i])
        { rh=hvs[i]; rs=svs[i]; rv=vvs[i]; cont=1; }
    else if(TempTls[i] && Temp [i+1])
        {
        rh=intrplcn(Tls[i],Tls[i+1],hvs[i],hvs[i+1],Temp);
        rs=intrplcn(Tls[i],Tls[i+1],svs[i],svs[i+1],Temp);
        rv=intrplcn(Tls[i],Tls[i+1],vvs[i],vvs[i+1],Temp);cont=1;
        }
    i++;
    }
}
/*=========================================*/
```

```
void vapor(double Temp,double entrop)          /* Estado vapor */
{
int i, cont, p1, p2, j, ri, rj;

cont=i=0;
while(i && cont==0)
    {
    if(Temp==Tv[i])
        { ri=i; cont=1; p1=1; }
    else if(TempTv[i] && Temp [i+1])
        { ri=i; cont=1; p1=2;}
    i++;
    }
if(p1==1)
    {
    j=cont=0;
    while(j && cont==0)
        {
        if(entrop==sv[ri][j])
            { rj=j; cont=1; p2=1; }
        else if(entropsv[ri][j] && entrop[ri][j+1])
            { rj=j; cont=1; p2=2; }
        j++;
        }
    }
else
    {
    j=cont=0;
    while(j && cont==0)
        {
        ins1=intrplcn(Tv[ri],Tv[ri+1],sv[ri][j],sv[ri+1][j],Temp);
        ins2=intrplcn(Tv[ri],Tv[ri+1],sv[ri][j+1],sv[ri+1][j+1],Temp);
        if(entrop==ins1)
            { rj=j; cont=1; p2=1; }
        else if(entropins1 && entrop 2)
            { rj=j; cont=1; p2=2; }
        j++;
        }
    }
ins1=intrplcn(Tv[ri],Tv[ri+1],sv[ri][rj],sv[ri+1][rj],Temp);
inh1=intrplcn(Tv[ri],Tv[ri+1],hv[ri][rj],hv[ri+1][rj],Temp);
inv1=intrplcn(Tv[ri],Tv[ri+1],vv[ri][rj],vv[ri+1][rj],Temp);
ins2=intrplcn(Tv[ri],Tv[ri+1],sv[ri][rj+1],sv[ri+1][rj+1],Temp);
inh2=intrplcn(Tv[ri],Tv[ri+1],hv[ri][rj+1],hv[ri+1][rj+1],Temp);
```

```
inv2=intrplcn(Tv[ri],Tv[ri+1],vv[ri][rj+1],vv[ri+1][rj+1],Temp);
if(p1==1 && p2==1)
    { rx=xv[rj]; rh=hv[ri][rj]; rv=vv[ri][rj]; }
else if(p1==1 && p2==2)
    {
    rx=intrplcn(sv[ri][rj],sv[ri][rj+1],xv[rj],xv[rj+1],entrop);
    rh=intrplcn(sv[ri][rj],sv[ri][rj+1],hv[ri][rj],hv[ri][rj+1],entrop);
    rv=intrplcn(sv[ri][rj],sv[ri][rj+1],vv[ri][rj],vv[ri][rj+1],entrop);
    }
else if(p1==2 && p2==1)
    { rx=xv[rj]; rh=inh1; rv=inv1;}
else {
    rx=intrplcn(ins1,ins2,xv[rj],xv[rj+1],entrop);
    rh=intrplcn(xv[rj],xv[rj+1],inh1,inh2,rx);
    rv=intrplcn(xv[rj],xv[rj+1],inv1,inv2,rx);
    }
}
/*=========================================*/
void vapsupercal(double Temp,double Pres)        /* Estado supercalentado */
{
int i, j, ri, rj, cont, n, p1, p2, rn;

cont=i=0;
while(i && cont==0)
    {
    if(Pres==Psh[i])
        { ri=i; cont=1; p1=1; }
    else if(Pres<Psh[i] && Pres>Psh[i+1])
        { ri=i; cont=1; p1=2;}
    i++;
    }
cont=n=0;
while(n && cont==0)
    {
    if(n==ri)
        { rn=n; cont=1; }
    n++;
    }
cont=0; j=a[rn];
while(j && cont==0)
    {
    if(Temp==Tsh[j])
        { rj=j; cont=1; p2=1; }
    else if(Temp<Tsh[j] && Temp>Tsh[j+1])
        { rj=j; cont=1; p2=2;}
    j++;
    }
if(p1==1 && p2==1)
    { rh=hsh[ri][rj]; rs=ssh[ri][rj]; }
else if(p1==1 && p2==2)
    {
    rh=intrplcn(Tsh[rj],Tsh[rj+1],hsh[ri][rj],hsh[ri][rj+1],Temp);
    rs=intrplcn(Tsh[rj],Tsh[rj+1],ssh[ri][rj],ssh[ri][rj+1],Temp);
    }
else if(p1==2 && p2==1)
    {
    rh=intrplcn(Psh[ri],Psh[ri+1],hsh[ri][rj],hsh[ri+1][rj],Pres);
    rs=intrplcn(Psh[ri],Psh[ri+1],ssh[ri][rj],ssh[ri+1][rj],Pres);
    }
else
    {
    rh=0.5*(0.5*(hsh[ri][rj]+hsh[ri+1][rj])+0.5*(hsh[ri][rj+1]+hsh[ri+1][rj+1]));
    rs=0.5*(0.5*(ssh[ri][rj]+ssh[ri+1][rj])+0.5*(ssh[ri][rj+1]+ssh[ri+1][rj+1]));
    }
}
/*=========================================*/
void vapsupercal2(double Pres,double entrop)
{
int i, cont, p1, p2, j, ri, rj;

cont=i=0;
while(i && cont==0)
    {
    if(Pres==Psh[i])
        { ri=i; cont=1; p1=1; }
    else if(Pres<Psh[i] && Pres>Psh[i+1])
        { ri=i; cont=1; p1=2; }
    i++;
    }
if(p1==1)
    {
    cont=0; j=a[ri];
    while(j && cont==0)
        {
        if(entrop==ssh[ri][j])
            { rj=j; cont=1; p2=1; }
        else if(entrop<ssh[ri][j] && entrop>ssh[ri][j+1])
            { rj=j; cont=1; p2=2; }
```

```
        j++;
        }
    }
else
    {
    cont=0; j=a[ri];
    while(j && cont==0)
        {
        ins1=intrplcn(Psh[ri],Psh[ri+1],ssh[ri][j],ssh[ri+1][j],Pres);
        ins2=intrplcn(Psh[ri],Psh[ri+1],ssh[ri][j+1],ssh[ri+1][j+1],Pres);
        if(entrop==ins1)
            { rj=j; cont=1; p2=1; }
        else if(entropins1 && entrop 2)
            { rj=j; cont=1; p2=2; }
        j++;
        }
    }
ins1=intrplcn(Psh[ri],Psh[ri+1],ssh[ri][rj],ssh[ri+1][rj],Pres);
inh1=intrplcn(Psh[ri],Psh[ri+1],hsh[ri][rj],hsh[ri+1][rj],Pres);
ins2=intrplcn(Psh[ri],Psh[ri+1],ssh[ri][rj+1],ssh[ri+1][rj+1],Pres);
inh2=intrplcn(Psh[ri],Psh[ri+1],hsh[ri][rj+1],hsh[ri+1][rj+1],Pres);
if(p1==1 && p2==1)
    { rt=Tsh[rj]; rh=hsh[ri][rj]; }
else if(p1==1 && p2==2)
    {
    rt=intrplcn(ssh[ri][rj],ssh[ri][rj+1],Tsh[rj],Tsh[rj+1],entrop);
    rh=intrplcn(ssh[ri][rj],ssh[ri][rj+1],hsh[ri][rj],hsh[ri][rj+1],entrop);
    }
else if(p1==2 && p2==1)
    { rt=Tsh[rj]; rh=inh1; }
else
    {
    rt=intrplcn(ins1,ins2,Tsh[rj],Tsh[rj+1],entrop);
    rh=intrplcn(Tsh[rj],Tsh[rj+1],inh1,inh2,rt);
    }
}
/*=========================================*/
double intrplcn(double x1,double x2,double y1,double y2,double x0)
{
doubleres_intrplcn=y1+(x0-x1)*(y2-y1)/(x2-x1);
return(res_intrplcn);
}
/*=========================================*/
```



```
double vapsat_return_h(double Temp)     /* Estado vapor saturado */
{
int i, cont;
double entrop;

cont=i=0;
while(i: && cont==0)
    {
    if(Temp==Tls[i])
        { entrop=hvs[i]; cont=1; }
    else if(TempTls[i] && Temp [i+1])
        {entrop=intrplcn(Tls[i],Tls[i+1],hvs[i],hvs[i+1],Temp); cont=1; }
    i++;
    }
return(entrop);
}
```

RANKGRAF.C

```
#include "rank.h"

extern int error_signal;
extern double Ta,Tb,Rt,Rp,Tsc,Ph;
extern double x[12],T[12],P[12],v[12],s[12],h[12];
/*=========================================*/
void graf_rend_rank(int noc)
{
char ejeR[10], ejeT[5]| Talt[5];
extern int xl, yl;
int x, y, x0, y0, xm, ym, Rm, rend, temp, i, xa, ya;
double R, Tbaj, Tbp, step, Rs, Ra;

Tbp=Tb; Rm=100;
abrirgraf();
x0=2*(xl+1)/8; y0=6*(yl+1)/8; xm=6*(xl+1)/8; ym=(yl+1)/8;
rectangle(0.25*x0,0.5*ym,3.75*x0,7.5*ym);
setfillstyle(1,9); floodfill(2,2,15);
line(x0,ym,x0,y0); line(x0,y0,xm,y0);
setcolor(14);
```

```
switch(noc)
    {
    case 1:  settextstyle(2,1,6); outtextxy(0.5*x0,ym,"Efficiency Es,s");
             settextstyle(2,0,5); outtextxy(1.5*x0,ym*1.5,"SIMPLE CYCLE (IDEAL)");
             break;
    case 2:  settextstyle(2,1,6); outtextxy(0.5*x0,ym,"Efficiency Es,a");
             settextstyle(2,0,5); outtextxy(1.5*x0,ym*1.5,"SIMPLE CYCLE (REAL)");
             break;
    case 3:  settextstyle(2,1,6); outtextxy(0.4*x0,ym,"Esup,s -----");
             settextstyle(2,1,6); outtextxy(0.6*x0,ym,"Esup,a.....");
             settextstyle(2,0,5); outtextxy(1.5*x0,ym*1.5,"SUPERHEATED CYCLE");
             break;
    case 4:  settextstyle(2,1,6); outtextxy(0.4*x0,ym,"Ereh,s -----");
             settextstyle(2,1,6); outtextxy(0.6*x0,ym,"Ereh,a.....");
             settextstyle(2,0,5); outtextxy(1.5*x0,ym*1.5,"REHEATED CYCLE");
             break;
    default: break;
    }
settextstyle(2,0,6); outtextxy(2.5*x0,6.5*ym,"Temperature Tb");
setcolor(6); settextstyle(1,0,2);
outtextxy(1.5*x0,0.7*ym,"GRAPHIC OF EFFICIENCY");
settextstyle(0,0,1); setcolor(15);
sprintf(Talt,"Ta=%-5.1f",Ta); outtextxy(2.5*x0,2*ym,Talt);
for(y=ym,rend=100;yy0;y=y+(y0-ym)/10,rend=rend-10)
    { /*Eje de rendimientos*/ sprintf(ejeR,"%4d-",rend); outtextxy(x0-35,y-3,ejeR); }
step=(xm-x0)/Ta;
for(x=x0,i=0;x;x=x0+step*25*++i)
    {   /*Eje de temperaturas Tb*/ temp=25*i; sprintf(ejeT,"%d",temp);
    outtextxy(x,y0,"I"); outtextxy(x,6.2*(yl+1)/8,ejeT); }
Tb=0.; calc(noc,Ta,Tb,Rt,Rp,Tsc,Ph);
xa=x0; Rs=-(h[6]-h[5]+h[1]-h[8])*100./fabs(h[5]-h[1]); ya=y0+Rs*(ym-y0)/Rm;
for(Tbaj=0.;Tbaja;Tbaj=Tbaj+1.)
    {   /*Representacion de rendimientos*/
    Tb=Tbaj;calc(noc,Ta,Tb,Rt,Rp,Tsc,Ph);
    x=x0+Tbaj*(xm-x0)/Ta;
    switch(noc)
        {
        case 1:  R=-(h[4]+h[1]-h[3]-h[5])*100./fabs(h[3]-h[1]);
                 y=y0+R*(ym-y0)/Rm; putpixel(x,y,15); break;
        case 2:  R=-(h[6]-h[4]+h[2]-h[7])*100./fabs(h[4]-h[2]);
                 y=y0+R*(ym-y0)/Rm; putpixel(x,y,15); break;
        case 3:  Ra=-(h[7]-h[5]+h[2]-h[8])*100./fabs(h[5]-h[2]);
                 y=y0+Ra*(ym-y0)/Rm; putpixel(x,y,15);
                 Rs=-(h[6]-h[5]+h[1]-h[8])*100./fabs(h[5]-h[1]);
                 y=y0+Rs*(ym-y0)/Rm; line(x,y,xa,ya); xa=x; ya=y;
                 break;
        case 4:  Ra=-(h[7]+h[2]+h[10]-h[5]-h[8]-h[11])*100./fabs(h[5]+h[8]-h[2]-h[7]);
                 y=y0+Ra*(ym-y0)/Rm; putpixel(x,y,15);
                 Rs=-(h[6]+h[9]+h[1]-h[5]-h[8]-h[11])*100./fabs(h[5]+h[8]-h[1]-h[6]);
                 y=y0+Rs*(ym-y0)/Rm; line(x,y,xa,ya); xa=x; ya=y;
                 break;
        default: break;
        }
    }
getch(); closegraph();
Tb=Tbp;
calc(noc,Ta,Tb,Rt,Rp,Tsc,Ph);
}
/*=========================================*/
void graf(int noc)
{
extern int xl, yl;
int x0, y0, xpl, i, x, y, xp, yp, y0p, cntx, cnty, aj, xpos, ypos, rad;
int x1s, y1s, x1a, y1a, x2, y2, x3, y3, x4, y4, x4s, y4s, x4a, y4a,
int x5s, y5s, x5, y5, x5a, y5a, x6, y6;
int x5p, y5p, x5pps, y5pps, x5ppa, y5ppa;
double ypl;
extern double Pls[], hls[], hvsa[];

abrirgraf();
x0=(xl+1)/8; y0=yl+1-(yl+1)/8;
switch(noc)
    {
    case 1:  xpl=3000; break;
    case 2:  xpl=3000; break;
    case 3:  xpl=4000; break;
    case 4:  xpl=4000; break;
    default: break;
    }
ypl=log10(100000);
xp=x0+hls[0]*(xl-x0)/xpl; yp=y0-log10(Pls[0])*y0/ypl;
for(i=1;iL.;i++)
    {
    x=x0+hls[i]*(xl-x0)/xpl; y=y0-log10(Pls[i])*y0/ypl;
    setcolor(9); line(x,y,xp,yp);
    xp=x; yp=y;
    }
xp=x0+hvsa[0]*(xl-x0)/xpl; yp=y0-log10(Pls[0])*y0/ypl;
```

```
for(i=1;iL;i++)
    {
    x=x0+hvsa[i]*(xl-x0)/xpl; y=y0-log10(Pls[i])*y0/ypl;
    setcolor(12);line(x,y,xp,yp);
    xp=x; yp=y;
    }
setcolor(15);
y0p=y0+19*yl/479; y0=y0p; /*ajustamos escala vertical*/
line(x0,y0,xl,y0); line(x0,y0,x0,0);
cnty=y0/6; aj=xl/80;
outtextxy(x0-aj*6,0," 1000-"); outtextxy(x0-aj*6,cnty," 100-");
outtextxy(x0-aj*6,cnty*2," 10-"); outtextxy(x0-aj*6,cnty*3," 1-");
outtextxy(x0-aj*6,cnty*4," 0.1-"); outtextxy(x0-aj*6,cnty*5," 0.01-");
outtextxy(x0-aj*6,cnty*6-1,"0.001-");
if(noc==3 || noc==4)      /*escala horizontal*/
    cntx=(xl-x0)/8;
else
    cntx=(xl-x0)/6;
for(i=0;i;i++)
    {
    outtextxy(x0+i*cntx,y0-2,"|");}
    outtextxy(x0,y0+10,"0");outtextxy(x0+cntx,y0+10,"500");
    outtextxy(x0+2*cntx,y0+10,"1000");outtextxy(x0+3*cntx,y0+10,"1500");
    outtextxy(x0+4*cntx,y0+10,"2000");outtextxy(x0+5*cntx,y0+10,"2500");
    outtextxy(x0+6*cntx,y0+10,"3000");
    xpos=x0+cntx/5; ypos=cnty/3; rad=ypos/5;
    switch(noc)
        {
        case 1:  setcolor(14); rad=1.5*ypos/5;
            x1s=x0+h[1]*(xl-x0)/xpl; y1s=y0-log10(P[1])*y0/ypl;
            x2=x0+h[2]*(xl-x0)/xpl; y2=y0-log10(P[2])*y0/ypl;
            x3=x0+h[3]*(xl-x0)/xpl; y3=y0-log10(P[3])*y0/ypl;
            x4s=x0+h[4]*(xl-x0)/xpl; y4s=y0-log10(P[4])*y0/ypl;
            x5=x0+h[5]*(xl-x0)/xpl; y5=y0-log10(P[5])*y0/ypl;
            line(x1s,y1s,x2,y2); line(x2,y2,x3,y3); line(x3,y3,x4s,y4s);
            line(x4s,y4s,x5,y5); line(x5,y5,x1s,y1s);
            setcolor(15);
            rectangle(xpos-rad,ypos-rad,xpos+rad,ypos+rad);
            outtextxy(xpos+rad,ypos," 1s Liquid(ideal)");
            setfillstyle(1,15); floodfill(xpos,ypos,15);
            copimag(xpos-rad,ypos-rad,xpos+rad,ypos+rad,x1s-rad,y1s-rad,0);
            outtextxy(x1s+rad,y1s+rad,"1s");
            setcolor(9);rectangle(xpos-rad,ypos*2-rad,xpos+rad,ypos*2+rad);
            outtextxy(xpos+rad,ypos*2," 2 saturated liquid");
            setfillstyle(1,9); floodfill(xpos,ypos*2,9);
            copimag(xpos-rad,ypos*2-rad,xpos+rad,ypos*2+rad,x2-rad,y2-rad,0);
            outtextxy(x2+rad,y2+rad,"2");
            setcolor(12);rectangle(xpos-rad,ypos*3-rad,xpos+rad,ypos*3+rad);
            outtextxy(xpos+rad,ypos*3," 3 saturated vapor");
            setfillstyle(1,12); floodfill(xpos,ypos*3,12);
            copimag(xpos-rad,ypos*3-rad,xpos+rad,ypos*3+rad,x3-rad,y3-rad,0);
            outtextxy(x3+rad,y3+rad,"3");
            setcolor(14);
            rectangle(xpos+cntx*2-rad,ypos-rad,xpos+cntx*2+rad,ypos+rad);
            outtextxy(xpos+cntx*2+rad,ypos," 4s Vapor(ideal)");
            setfillstyle(1,14); floodfill(xpos+cntx*2,ypos,14);
            copimag(xpos+cntx*2-rad,ypos-rad,xpos+cntx*2+rad,ypos+rad,x4s-rad,y4s-rad,0);
            outtextxy(x4s+rad,y4s+rad,"4s");setcolor(9);
            rectangle(xpos+cntx*2-rad,ypos*2-rad,xpos+cntx*2+rad,ypos*2+rad);
            outtextxy(xpos+cntx*2+rad,ypos*2," 5 saturated liquid");
            setfillstyle(1,9); floodfill(xpos+cntx*2,ypos*2,9);
            copimag(xpos+cntx*2-rad,ypos*2-rad,xpos+cntx*2+rad,ypos*2+rad,x5-rad,y5-rad,0);
            outtextxy(x5+rad,y5+rad,"5");          break;
    case 2:  setcolor(14);
        x1s=x0+h[1]*(xl-x0)/xpl; y1s=y0-log10(P[1])*y0/ypl;
        x1a=x0+h[2]*(xl-x0)/xpl; y1a=y0-log10(P[2])*y0/ypl;
        x2=x0+h[3]*(xl-x0)/xpl; y2=y0-log10(P[3])*y0/ypl;
        x3=x0+h[4]*(xl-x0)/xpl; y3=y0-log10(P[4])*y0/ypl;
        x4s=x0+h[5]*(xl-x0)/xpl; y4s=y0-log10(P[5])*y0/ypl;
        x4a=x0+h[6]*(xl-x0)/xpl; y4a=y0-log10(P[6])*y0/ypl;
        x5=x0+h[7]*(xl-x0)/xpl; y5=y0-log10(P[7])*y0/ypl;
        line(x1s,y1s,x2,y2); line(x2,y2,x3,y3); line(x3,y3,x4s,y4s);
        line(x4s,y4s,x5,y5); line(x5,y5,x1s,y1s);
        setlinestyle(DOTTED_LINE,0,NORM_WIDTH);
        line(x3,y3,x4a,y4a); line(x4s,y4s,x4a,y4a); line(x5,y5,x1a,y1a);
        setlinestyle(SOLID_LINE,0,NORM_WIDTH); setcolor(WHITE);
        rectangle(xpos-rad,ypos-rad,xpos+rad,ypos+rad);
        outtextxy(xpos+rad,ypos," 1s Liquid");
        setfillstyle(1,15); floodfill(xpos,ypos,15);
        copimag(xpos-rad,ypos-rad,xpos+rad,ypos+rad,x1s-rad,y1s-rad,0);
        outtextxy(x1s+rad,y1s+rad,"1s");setcolor(LIGHTBLUE);
        rectangle(xpos-rad,ypos*2-rad,xpos+rad,ypos*2+rad);
        outtextxy(xpos+rad,ypos*2," 2 saturated liquid");
        setfillstyle(1,9); floodfill(xpos,ypos*2,9);
        copimag(xpos-rad,ypos*2-rad,xpos+rad,ypos*2+rad,x2-rad,y2-rad,0);
        outtextxy(x2+rad,y2+rad,"2");setcolor(LIGHTRED);
        rectangle(xpos-rad,ypos*3-rad,xpos+rad,ypos*3+rad);
        outtextxy(xpos+rad,ypos*3," 3 saturated vapor");
```

```
        setfillstyle(1,12); floodfill(xpos,ypos*3,12);
        copimag(xpos-rad,ypos*3-rad,xpos+rad,ypos*3+rad,x3-rad,y3-rad,0);
        outtextxy(x3+rad,y3+rad,"3"); setcolor(YELLOW);
        rectangle(xpos+cntx*2-rad,ypos-rad,xpos+cntx*2+rad,ypos+rad);
        outtextxy(xpos+cntx*2+rad,ypos," 4s Vapor(ideal)");
        setfillstyle(1,14); floodfill(xpos+cntx*2,ypos,14);
        copimag(xpos+cntx*2-rad,ypos-rad,xpos+cntx*2+rad,ypos+rad,x4s-rad,y4s-rad,0);
        outtextxy(x4s+rad,y4s+rad,"4s"); setcolor(BROWN);
        rectangle(xpos+cntx*2-rad,ypos*2-rad,xpos+cntx*2+rad,ypos*2+rad);
        outtextxy(xpos+cntx*2+rad,ypos*2," 4a Vapor(real)");
        setfillstyle(1,6); floodfill(xpos+cntx*2,ypos*2,6);
        copimag(xpos+cntx*2-rad,ypos*2-rad,xpos+cntx*2+rad,ypos*2+rad,x4a-rad,y4a-rad,0);
        outtextxy(x4a+rad,y4a+rad,"4a"); setcolor(LIGHTBLUE);
        rectangle(xpos+cntx*4-rad,ypos-rad,xpos+cntx*4+rad,ypos+rad);
        outtextxy(xpos+cntx*4+rad,ypos," 5 saturated liquid");
        setfillstyle(1,9); floodfill(xpos+cntx*4,ypos,9);
        copimag(xpos+cntx*4-rad,ypos-rad,xpos+cntx*4+rad,ypos+rad,x5-rad,y5-rad,0);
        outtextxy(x5+rad,y5+rad,"5"); break;
case 3: setcolor(14);
        x1s=x0+h[1]*(xl-x0)/xpl;    y1s=y0-log10(P[1])*y0/ypl;
        x1a=x0+h[2]*(xl-x0)/xpl;    y1a=y0-log10(P[2])*y0/ypl;
        x2=x0+h[3]*(xl-x0)/xpl;     y2=y0-log10(P[3])*y0/ypl;
        x3=x0+h[4]*(xl-x0)/xpl;     y3=y0-log10(P[4])*y0/ypl;
        x4=x0+h[5]*(xl-x0)/xpl;     y4=y0-log10(P[5])*y0/ypl;
        x5s=x0+h[6]*(xl-x0)/xpl;    y5s=y0-log10(P[6])*y0/ypl;
        x5a=x0+h[7]*(xl-x0)/xpl;    y5a=y0-log10(P[7])*y0/ypl;
        x6=x0+h[8]*(xl-x0)/xpl;     y6=y0-log10(P[8])*y0/ypl;
        line(x1s,y1s,x2,y2);    line(x2,y2,x3,y3);    line(x3,y3,x4,y4);
        line(x4,y4,x5s,y5s);    line(x5s,y5s,x6,y6);  line(x6,y6,x1s,y1s);
        setlinestyle(DOTTED_LINE,0,NORM_WIDTH);
        line(x4,y4,x5a,y5a);    line(x5s,y5s,x5a,y5a); line(x6,y6,x1a,y1a);
        setlinestyle(SOLID_LINE,0,NORM_WIDTH);
        setcolor(WHITE);rectangle(xpos-rad,ypos-rad,xpos+rad,ypos+rad);
        outtextxy(xpos+rad,ypos," 1 Liquid");
        setfillstyle(1,15); floodfill(xpos,ypos,15);
        copimag(xpos-rad,ypos-rad,xpos+rad,ypos+rad,x1s-rad,y1s-rad,0);
        outtextxy(x1s+rad,y1s+rad,"1s");setcolor(LIGHTBLUE);
        rectangle(xpos-rad,ypos*2-rad,xpos+rad,ypos*2+rad);
        outtextxy(xpos+rad,ypos*2," 2 saturated liquid");
        setfillstyle(1,9); floodfill(xpos,ypos*2,9);
        copimag(xpos-rad,ypos*2-rad,xpos+rad,ypos*2+rad,x2-rad,y2-rad,0);
        outtextxy(x2+rad,y2+rad,"2"); setcolor(LIGHTRED);
        rectangle(xpos-rad,ypos*3-rad,xpos+rad,ypos*3+rad);
        outtextxy(xpos+rad,ypos*3," 3 saturated vapor");
        setfillstyle(1,12); floodfill(xpos,ypos*3,12);
        copimag(xpos-rad,ypos*3-rad,xpos+rad,ypos*3+rad,x3-rad,y3-rad,0);
        outtextxy(x3+rad,y3+rad,"3"); setcolor(BROWN);
        rectangle(xpos-rad,ypos*4-rad,xpos+rad,ypos*4+rad);
        outtextxy(xpos+rad,ypos*4," 4 superheat.vapor");
        setfillstyle(1,6); floodfill(xpos,ypos*4,6);
        copimag(xpos-rad,ypos*4-rad,xpos+rad,ypos*4+rad,x4-rad,y4-rad,0);
        outtextxy(x4+rad,y4+rad,"4"); setcolor(YELLOW);
        rectangle(xpos+cntx*2.5-rad,ypos-rad,xpos+cntx*2.5+rad,ypos+rad);
        outtextxy(xpos+cntx*2.5+rad,ypos," 5s Vapor(ideal)");
        setfillstyle(1,14); floodfill(xpos+cntx*2.5,ypos,14);
        copimag(xpos+cntx*2.5-rad,ypos-rad,xpos+cntx*2.5+rad,ypos+rad,x5s-rad,y5s-rad,0);
        outtextxy(x5s+rad,y5s+rad,"5s");setcolor(YELLOW);
        rectangle(xpos+cntx*2.5-rad,ypos*2-rad,xpos+cntx*2.5+rad,ypos*2+rad);
        outtextxy(xpos+cntx*2.5+rad,ypos*2," 5a Vapor(real)");
        setfillstyle(1,14); floodfill(xpos+cntx*2.5,ypos*2,14);
        copimag(xpos+cntx*2.5-rad,ypos*2-rad,xpos+cntx*2.5+rad,ypos*2+rad,x5a-rad,y5a-rad,0);
        outtextxy(x5a+rad,y5a+rad,"5a");setcolor(LIGHTBLUE);
        rectangle(xpos+cntx*5-rad,ypos-rad,xpos+cntx*5+rad,ypos+rad);
        outtextxy(xpos+cntx*5+rad,ypos," 6 saturated liquid");
        setfillstyle(1,9); floodfill(xpos+cntx*5,ypos,9);
        copimag(xpos+cntx*5-rad,ypos-rad,xpos+cntx*5+rad,ypos+rad,x6-rad,y6-rad,0);
        outtextxy(x6+rad,y6+rad,"6");          break;
case 4: setcolor(14);
        x1s=x0+h[1]*(xl-x0)/xpl;    y1s=y0-log10(P[1])*y0/ypl;
        x1a=x0+h[2]*(xl-x0)/xpl;    y1a=y0-log10(P[2])*y0/ypl;
        x2=x0+h[3]*(xl-x0)/xpl;     y2=y0-log10(P[3])*y0/ypl;
        x3=x0+h[4]*(xl-x0)/xpl;     y3=y0-log10(P[4])*y0/ypl;
        x4=x0+h[5]*(xl-x0)/xpl;     y4=y0-log10(P[5])*y0/ypl;
        x5s=x0+h[6]*(xl-x0)/xpl;    y5s=y0-log10(P[6])*y0/ypl;
        x5a=x0+h[7]*(xl-x0)/xpl;    y5a=y0-log10(P[7])*y0/ypl;
        x5p=x0+h[8]*(xl-x0)/xpl;    y5p=y0-log10(P[8])*y0/ypl;
        x5pps=x0+h[9]*(xl-x0)/xpl;  y5pps=y0-log10(P[9])*y0/ypl;
        x5ppa=x0+h[10]*(xl-x0)/xpl;y5ppa=y0-log10(P[10])*y0/ypl;
        x6=x0+h[11]*(xl-x0)/xpl;    y6=y0-log10(P[11])*y0/ypl;
        line(x1s,y1s,x2,y2);    line(x2,y2,x3,y3);    line(x3,y3,x4,y4);
        line(x4,y4,x5s,y5s);    line(x5s,y5s,x5p,y5p); line(x5p,y5p,x5pps,y5pps);
        line(x5pps,y5pps,x6,y6); line(x6,y6,x1s,y1s);
        setlinestyle(DOTTED_LINE,0,NORM_WIDTH);
        line(x4,y4,x5a,y5a);    line(x5p,y5p,x5ppa,y5ppa);
        line(x5ppa,y5ppa,x5pps,y5pps);
        setlinestyle(SOLID_LINE,0,NORM_WIDTH); setcolor(WHITE);
        rectangle(xpos-rad,ypos-rad,xpos+rad,ypos+rad);
        outtextxy(xpos+rad,ypos,"1 Liqid");
```

```
        setfillstyle(1,15); floodfill(xpos,ypos,15);
        copimag(xpos-rad,ypos-rad,xpos+rad,ypos+rad,x1s-rad,y1s-rad,0);
        outtextxy(x1s+rad,y1s+rad,"1s");setcolor(LIGHTBLUE);
        rectangle(xpos-rad,ypos*2-rad,xpos+rad,ypos*2+rad);
        outtextxy(xpos+rad,ypos*2," 2 saturated liquid");
        setfillstyle(1,9); floodfill(xpos,ypos*2,9);
        copimag(xpos-rad,ypos*2-rad,xpos+rad,ypos*2+rad,x2-rad,y2-rad,0);
        outtextxy(x2+rad,y2+rad,"2");setcolor(LIGHTRED);
        rectangle(xpos-rad,ypos*3-rad,xpos+rad,ypos*3+rad);
        outtextxy(xpos+rad,ypos*3," 3 saturated vapor");
        setfillstyle(1,12); floodfill(xpos,ypos*3,12);
        copimag(xpos-rad,ypos*3-rad,xpos+rad,ypos*3+rad,x3-rad,y3-rad,0);
        outtextxy(x3+rad,y3+rad,"3");setcolor(BROWN);
        rectangle(xpos-rad,ypos*4-rad,xpos+rad,ypos*4+rad);
        outtextxy(xpos+rad,ypos*4," 4 Superheat.(high press.)");
        setfillstyle(1,6); floodfill(xpos,ypos*4,6);
        copimag(xpos-rad,ypos*4-rad,xpos+rad,ypos*4+rad,x4-rad,y4-rad,0);
        outtextxy(x4+rad,y4+rad,"4");setcolor(GREEN);
        rectangle(xpos+cntx*2.5-rad,ypos-rad,xpos+cntx*2.5+rad,ypos+rad);
        outtextxy(xpos+cntx*2.5+rad,ypos," 5s Sup.(ideal)");
        setfillstyle(1,2); floodfill(xpos+cntx*2.5,ypos,2);
        copimag(xpos+cntx*2.5-rad,ypos-rad,xpos+cntx*2.5+rad,ypos+rad,x5s-rad,y5s-rad,0);
        outtextxy(x5s+rad,y5s+rad,"5s");setcolor(CYAN);
        rectangle(xpos+cntx*2.5-rad,ypos*2-rad,xpos+cntx*2.5+rad,ypos*2+rad);
        outtextxy(xpos+cntx*2.5+rad,ypos*2," 5a Sup.(real)");
        setfillstyle(1,3); floodfill(xpos+cntx*2.5,ypos*2,3);
copimag(xpos+cntx*2.5-rad,ypos*2-rad,xpos+cntx*2.5+rad,ypos*2+rad,x5a-rad,y5a-rad,0);
        outtextxy(x5a+rad,y5a+rad,"5a");setcolor(BROWN);
        rectangle(xpos+cntx*5-rad,ypos-rad,xpos+cntx*5+rad,ypos+rad);
        outtextxy(xpos+cntx*5+rad,ypos," 5' Reheated");
        setfillstyle(1,6); floodfill(xpos+cntx*5,ypos,6);
        copimag(xpos+cntx*5-rad,ypos-rad,xpos+cntx*5+rad,ypos+rad,x5p-rad,y5p-rad,0);
        outtextxy(x5p+rad,y5p+rad,"5'");setcolor(YELLOW);
        rectangle(xpos+cntx*5-rad,ypos*2-rad,xpos+cntx*5+rad,ypos*2+rad);
        outtextxy(xpos+cntx*5+rad,ypos*2," 5''s Vapor(ideal)");
        setfillstyle(1,14); floodfill(xpos+cntx*5,ypos*2,14);
copimag(xpos+cntx*5-rad,ypos*2-rad,xpos+cntx*5+rad,ypos*2+rad,x5pps-rad,y5pps-rad,0);
        outtextxy(x5pps+rad,y5pps+rad,"5''s");setcolor(YELLOW);
        rectangle(xpos+cntx*5-rad,ypos*3-rad,xpos+cntx*5+rad,ypos*3+rad);
        outtextxy(xpos+cntx*5+rad,ypos*3," 5''a Vapor(real)");
        setfillstyle(1,14); floodfill(xpos+cntx*5,ypos*3,14);
copimag(xpos+cntx*5-rad,ypos*3-rad,xpos+cntx*5+rad,ypos*3+rad,x5ppa-rad,y5ppa-rad,0);
        outtextxy(x5ppa+rad,y5ppa+rad,"5''a");setcolor(LIGHTBLUE);
        rectangle(xpos+cntx*5-rad,ypos*4-rad,xpos+cntx*5+rad,ypos*4+rad);
        outtextxy(xpos+cntx*5+rad,ypos*4," 6' saturated liquid");
        setfillstyle(1,9); floodfill(xpos+cntx*5,ypos*4,9);
        copimag(xpos+cntx*5-rad,ypos*4-rad,xpos+cntx*5+rad,pos*4+rad,x6-rad,y6-rad,0);
        outtextxy(x6+rad,y6+rad,"6"); break;
    default: break;
    }
settextstyle(1,1,1); outtextxy(x0/3,y0/10,"Pressure (bar)");
settextstyle(1,0,1); outtextxy(x0*5,y0+18,"Entalpy (kJ/kg)");
settextstyle(0,0,0);
getch(); closegraph();
}
```

EXPLN1.C

```
#include "rank.h"
/*=========================================*/
void exp_is(void)
{
char tecla;

do  {
    do  {
        textrankis1();
        tecla=toupper(getch());
        switch(tecla)
            {
            case 'S':   esquema(1); break;
            case 'D':   diagram(1); break;
            case 'N':   break;
            default: puts("\a");
            }
        }
        while(tecla!='N');
    do  {
        textrankis2();
        tecla=toupper(getch());
        switch(tecla)
            {
            case 'S':   esquema(1); break;
            case 'D':   diagram(1); break;
```

```
            case 'P':     break;
            case 'X':     break;
            default: puts("\a");
            }
        }
        while(tecla=='S' || tecla=='D');
    }
    while(tecla=='P');
}
/*=========================================*/
void textrankis1(void)
{
clrscr(); textmode(C80); textcolor(15); textbackground(9);
cputs("                IDEAL AND SIMPLE RANKINE CYCLE                ");
cputs("The simple Rankine cycle uses water as a working fluid. It has five states.");
cputs("Heat transfer takes place in a constant-pressure process in the boiler. Liquid");
cputs("enters the boiler from the pump at low temperature (state 1: LIQUID) and is");
cputs("heated to saturation along (1 - 2). The liquid increases sligthly in volume");
cputs("(thermal expansion). At state 2 (SATURATED LIQUID),the saturation tem-");
cputs("perature is reached. Since the boiler is operated at effectively constant pre-");
cputs("ssure,further heat transfer takes place at const. temperature.And the energy");
cputs("added to the water goes into the heat of vaporization, producing SATURATED VAPOR");
cputs("(state 3). Then the vapor is expanded (in an assumed isentropic process for the");
cputs("simple cycle) through a turbine to produce work and leaves at state 4 (VAPOR).");
cputs("The expansion is limited in the practical cycle by the appearance of vapor con-");
cputs("densation in the turbine and by the saturation press. available at the tempe-");
cputs("rature of the cooling medium used in the condenser. If excessive condensation is");
cputs("allowed to occur, the condensed liquid droplets will rapidly erode the turbine");
cputs("blades. At the end of the expansion process (state 4), low-pressure but fairly");
cputs("high-quality vapor leaves the turbine. The vapor is condensed to a liquid");
cputs("as the vapor comes into contact with surfaces in the condenser that are cooled,");
cputs("usually by a cold-water stream. Because the condenser operates near the tempera-");
cputs("ture of the cooling water, the condensation process occurs at temperatures well");
cputs("below the normal (atmospheric) boiling point of most working fluids. The liquid");
cputs("*************************************************************");
cputs("* Options: D == DIAGRAMS S == SCHEME N == NEXT PAGE         *");
cputs("*************************************************************");
}
/*=========================================*/
void textrankis2(void)
{
textmode(C80); clrscr(); textcolor(15); textbackground(9);
cputs("leaves the condenser at state 5: SATURATED LIQUID. Following condensation, the");
cputs("liquid enters into a pump. The working fluid is returned to the high pressure");
cputs("necessary for energy addition at the higher boiler temperature.          ");
cputs("    And now,the cycle is repeated.                                      ");
cputs("                                                                        ");
cputs("                                                                        ");
cputs("                    EFFICIENCY CALCULUS                                 ");
cputs("                                                                        ");
cputs("                          W(3-4) + W(5-1)                               ");
cputs(" E = efficiency (of the cycle) = - ----------------                     ");
cputs("                          |Q(1-3)|                                     ");
cputs("                                                                        ");
cputs("W(3-4)=h4-h3              Vapor work produced in the turbine            ");
cputs("W(5-1)=h1-h5              Pump work produced on the water               ");
cputs("Q(1-3)=Q(1-2)+Q(2-3)=(h2-h1) + (h3-h2)    Heat absorbed by water      ");
cputs("                                                                        ");
cputs("h   entalpy (kJ/kg)          s   entropy (kJ/(kg.'K))                  ");
cputs("v   specific volume (m3/kg)      P   pressure (kPa)                     ");
cputs("T   temperature('k)          Q,W  heat,work  (kJ/kg)                   ");
cputs("*************************************************************");
cputs("*   D=DIAGRAMS  S=SCHEME  P=PREVIOUS PAGE   X=EXIT   *");
cputs("*************************************************************");
}
/*=========================================*/
void esquema(int num)
{
extern int xl,yl;
int u, v;
abrirgraf();
u=(xl+1)/80; v=(yl+1)/58;
rectangle(2*u,2*v,78*u,57*v); rectangle(3*u,3*v,77*u,56*v);      /*marcos*/
setfillstyle(1,9); floodfill(1,1,15);
setfillstyle(1,14); floodfill(2*u+1,2*v+1,15);
rectangle(8*u,33*v,13*u,38*v);                                   /*bomba*/
setfillstyle(6,2); floodfill(10*u,35*v,15);
moveto(8*u,33*v); lineto(8*u,23*v); lineto(11*u,23*v);   /*liquido y caldera*/
lineto(11*u,18*v); lineto(12*u,17*v); lineto(13*u,18*v); lineto(14*u,17*v);
lineto(15*u,18*v); lineto(16*u,17*v); lineto(17*u,18*v); lineto(18*u,17*v);
lineto(19*u,18*v); lineto(19*u,26*v); lineto(11*u,26*v); lineto(11*u,24*v);
lineto(9*u,24*v); lineto(9*u,33*v);
circle(16*u,20*v,7); circle(13*u,21*v,7); circle(17*u,23*v,7);  /*liquido sat*/
setfillstyle(1,9); floodfill(12*u,18*v,15);
setfillstyle(9,1); floodfill(16*u,20*v,15);
setfillstyle(9,1); floodfill(13*u,21*v,15);
setfillstyle(9,1); floodfill(17*u,23*v,15);
moveto(11*u,18*v); lineto(11*u,13*v); lineto(25*u,13*v);    /*vapor saturado*/
```

```
outtextxy(20*u,22*v,"saturated"); outtextxy(20*u,23*v,"liquid(2)");
line(17*u,23*v,20*u,23*v);line(17*u,14*v,12*u,9*v);
outtextxy(12*u,8*v,"saturated vapor(3)"); outtextxy(5*u,39*v,"PUMP");
outtextxy(18*u,31*v,"BURNERS");line(18*u,32*v,16*u,30*v);
line(14*u,40*v,18*u,36*v);outtextxy(30*u,32*v,"CONDENSER");
outtextxy(49*u,31*v,"Input cooling water");
outtextxy(43*u,42*v,"Output cooling water"); line(53*u,37*v,55*u,35*v);
switch (num)
    {
    case 1:  moveto(25*u,13*v); lineto(67*u,13*v); lineto(67*u,20*v);
             lineto(66*u,20*v); lineto(66*u,14*v); lineto(24*u,14*v);
             setfillstyle(6,4); floodfill(12*u,14*v,15);
             rectangle(12*u,29*v,18*u,30*v);                    /*hogar*/
             setfillstyle(1,6); floodfill(12*u+1,29*v+1,15);
             copimag(12*u,27*v,14*u,28*v,15*u,27*v,0);          /*fuego*/
             copimag(20*u,10*v,22*u,12*v,46*u,10*v,0);          /*flechas*/
             outtextxy(18*u,35*v,"saturatedliquid(5)");
             outtextxy(55*u,34*v,"Vapor(4)");
             outtextxy(69*u,14*v,"TURBINE");
             break;
    case 3:  moveto(25*u,13*v); lineto(25*u,21*v); lineto(40*u,21*v);
             lineto(40*u,20*v); lineto(29*u,20*v); lineto(29*u,13*v);
             lineto(67*u,13*v); lineto(67*u,20*v); lineto(66*u,20*v);
             lineto(66*u,14*v); lineto(30*u,14*v); lineto(30*u,19*v);
             lineto(41*u,19*v); lineto(41*u,22*v); lineto(24*u,22*v);
             lineto(24*u,14*v);
             setfillstyle(6,4); floodfill(12*u,14*v,15);
             moveto(27*u,21*v); lineto(27*u,11*v); lineto(42*u,11*v);
             lineto(42*u,13*v); moveto(42*u,14*v); lineto(42*u,24*v);
             lineto(27*u,24*v); lineto(27*u,22*v); setfillstyle(8,2);
             floodfill(28*u,12*v,15); floodfill(41*u,23*v,15);
             moveto(12*u,29*v); lineto(22*u,29*v);              /*hogar*/
             lineto(22*u,27*v); lineto(39*u,27*v); lineto(39*u,28*v);
             lineto(23*u,28*v); lineto(23*u,30*v); lineto(12*u,30*v);
             lineto(12*u,29*v);
             setfillstyle(1,6); floodfill(12*u+1,29*v+1,15);
             copimag(12*u,27*v,14*u,28*v,15*u,27*v,0);          /*fuego*/
             copimag(12*u,27*v,14*u,28*v,28*u,25*v,0);
             copimag(12*u,27*v,14*u,28*v,31*u,25*v,0);
             copimag(12*u,27*v,14*u,28*v,34*u,25*v,0);
             copimag(12*u,27*v,14*u,28*v,37*u,25*v,0);
             copimag(20*u,10*v,22*u,12*v,46*u,10*v,0);          /*flechas*/
             copimag(71*u,33*v,73*u,35*v,21*u,15*v,0);
             outtextxy(18*u,35*v,"saturatedliquid(6)");
```

```
moveto(19*u,18*v); lineto(19*u,14*v); lineto(24*u,14*v);
moveto(66*u,20*v); lineto(67*u,20*v); lineto(70*u,17*v);      /*turbina low*/
lineto(70*u,26*v); lineto(69*u,26*v); lineto(67*u,24*v); lineto(66*u,24*v);
lineto(66*u,20*v); setfillstyle(4,9); floodfill(68*u,22*v,15);
moveto(13*u,38*v); lineto(13*u,40*v); lineto(30*u,40*v);      /*liquid.sat 5*/
lineto(33*u,37*v); lineto(35*u,37*v); lineto(37*u,35*v); lineto(45*u,35*v);
lineto(45*u,36*v); lineto(37*u,36*v); lineto(36*u,37*v); lineto(42*u,37*v);
lineto(42*u,38*v); lineto(36*u,38*v); lineto(37*u,39*v); lineto(39*u,39*v);
lineto(39*u,40*v); lineto(37*u,40*v); lineto(35*u,38*v); lineto(33*u,38*v);
lineto(30*u,41*v); lineto(12*u,41*v); lineto(12*u,38*v);
setfillstyle(1,1); floodfill(13*u+1,40*v+1,15);
moveto(70*u,26*v); lineto(70*u,38*v); lineto(50*u,38*v);          /*vapor*/
lineto(48*u,40*v); lineto(39*u,40*v); lineto(39*u,39*v); lineto(48*u,39*v);
lineto(49*u,38*v); lineto(42*u,38*v); lineto(42*u,37*v); lineto(49*u,37*v);
lineto(48*u,36*v); lineto(45*u,36*v); lineto(45*u,35*v); lineto(48*u,35*v);
lineto(50*u,37*v); lineto(69*u,37*v); lineto(69*u,26*v);
setfillstyle(9,12); floodfill(69*u+1,37*v+1,15);
moveto(34*u,37*v); lineto(34*u,34*v); lineto(43*u,34*v);      /*condensador*/
lineto(43*u,33*v); lineto(45*u,33*v); lineto(46*u,34*v); lineto(47*u,33*v);
lineto(49*u,33*v); lineto(49*u,34*v); lineto(51*u,34*v); lineto(51*u,37*v);
lineto(51*u,37*v); moveto(34*u,38*v); lineto(34*u,41*v); lineto(36*u,41*v);
lineto(36*u,43*v); lineto(38*u,43*v); lineto(39*u,44*v); lineto(40*u,43*v);
lineto(42*u,43*v); lineto(42*u,41*v); lineto(51*u,41*v); lineto(51*u,38*v);
setfillstyle(6,6); floodfill(35*u,40*v,15); floodfill(34*u+1,34*v+1,15);
floodfill(37*u+1,36*v+1,15);floodfill(37*u,38*v+1,15);
moveto(46*u,34*v); lineto(44*u,32*v); lineto(45*u,32*v);      /*agua refrig*/
lineto(45*u,31*v); lineto(47*u,31*v); lineto(47*u,32*v); lineto(48*u,32*v);
lineto(46*u,34*v); moveto(39*u,44*v); lineto(37*u,42*v); lineto(38*u,42*v);
lineto(38*u,41*v); lineto(40*u,41*v); lineto(40*u,42*v); lineto(41*u,42*v);
lineto(39*u,44*v);
setfillstyle(1,9); floodfill(39*u,43*v,15); floodfill(46*u,33*v,15);
moveto(12*u,28*v); lineto(13*u,27*v); lineto(14*u,28*v);          /*fuego*/
lineto(12*u,28*v); setfillstyle(1,14); floodfill(13*u,27*v+1,15);
line(5*u,28*v,6*u,27*v); line(7*u,28*v,6*u,27*v);             /*flecha up*/
line(6*u,29*v,6*u,27*v);
line(21*u,10*v,22*u,11*v); line(22*u,11*v,20*u,11*v);         /*flecha right*/
line(22*u,11*v,21*u,12*v);
line(71*u,34*v,72*u,35*v); line(72*u,33*v,72*u,35*v);         /*flecha down*/
line(73*u,34*v,72*u,35*v);
line(58*u,40*v,59*u,39*v); line(58*u,40*v,60*u,40*v);         /*flecha left*/
line(58*u,40*v,59*u,41*v);
copimag(58*u,39*v,60*u,41*v,16*u,42*v,0);
settextstyle(2,0,4); outtextxy(4*u,12*v,"BOILER");            /*textos*/
outtextxy(4*u,18*v,"Liquid(1)");line(4*u,19*v,8*u,23*v);
```

```
        outtextxy(55*u,34*v,"Vapor(5)");
        outtextxy(69*u,14*v,"TURBINE");
        break;
case 4: moveto(25*u,13*v); lineto(25*u,21*v); lineto(40*u,21*v);
        lineto(40*u,20*v); lineto(29*u,20*v); lineto(29*u,13*v);
        lineto(54*u,13*v); lineto(54*u,20*v); lineto(53*u,20*v);
        lineto(53*u,14*v); lineto(30*u,14*v); lineto(30*u,19*v);
        lineto(41*u,19*v); lineto(41*u,22*v); lineto(24*u,22*v);
        lineto(24*u,14*v);
        moveto(27*u,21*v); lineto(27*u,11*v); lineto(42*u,11*v);
        lineto(42*u,13*v); moveto(42*u,14*v); lineto(42*u,15*v);
        moveto(42*u,16*v); lineto(42*u,17*v); moveto(42*u,18*v);
        lineto(42*u,24*v); lineto(27*u,24*v); lineto(27*u,22*v);
        moveto(56*u,26*v); lineto(56*u,27*v); lineto(49*u,27*v);
        lineto(49*u,15*v); lineto(31*u,15*v); lineto(31*u,18*v);
        lineto(46*u,18*v) ;lineto(46*u,30*v); lineto(67*u,30*v);
        lineto(67*u,24*v); moveto(66*u,24*v); lineto(66*u,29*v);
        lineto(47*u,29*v); lineto(47*u,17*v) ;lineto(32*u,17*v);
        lineto(32*u,16*v); lineto(48*u,16*v); lineto(48*u,28*v);
        lineto(57*u,28*v); lineto(57*u,26*v);
        moveto(53*u,20*v) ;lineto(54*u,20*v);          /*turbina high*/
        lineto(57*u,17*v); lineto(57*u,26*v); lineto(56*u,26*v);
        lineto(54*u,24*v); lineto(53*u,24*v); lineto(53*u,20*v);
        setfillstyle(4,9); floodfill(55*u,22*v,15);
        setfillstyle(9,8); floodfill(31*u+1,15*v+1,15);
        setfillstyle(6,4); floodfill(12*u,14*v,15); setfillstyle(8,2); floodfill(28*u,12*v,15);
        floodfill(41*u,23*v,15); floodfill(34*u+1,16*v+1,15);
        line(57*u,21*v,66*u,21*v);line(57*u,22*v,66*u,22*v);
        floodfill(60*u,21*v+1,15);
        moveto(12*u,29*v);lineto(22*u,29*v);              /*hogar*/
        lineto(22*u,27*v); lineto(39*u,27*v); lineto(39*u,28*v);
        lineto(23*u,28*v); lineto(23*u,30*v); lineto(12*u,30*v);
        lineto(12*u,29*v); setfillstyle(1,6); floodfill(12*u+1,29*v+1,15);
        copimag(12*u,27*v,14*u,28*v,15*u,27*v,0);          /*fuego*/
        copimag(12*u,27*v,14*u,28*v,28*u,25*v,0);
        copimag(12*u,27*v,14*u,28*v,31*u,25*v,0);
        copimag(12*u,27*v,14*u,28*v,34*u,25*v,0);
        copimag(12*u,27*v,14*u,28*v,37*u,25*v,0);
        copimag(20*u,10*v,22*u,12*v,46*u,10*v,0);          /*flechas*/
        copimag(20*u,10*v,22*u,12*v,62*u,31*v,0);
        copimag(5*u,25*v,7*u,29*v,63*u,24*v,0);
        copimag(5*u,25*v,7*u,29*v,50*u,22*v,0);
        copimag(71*u,33*v,73*u,35*v,43*u,19*v,0);
        copimag(71*u,33*v,73*u,35*v,55*u,13*v,0);
        copimag(71*u,33*v,73*u,35*v,21*u,15*v,0);
        outtextxy(18*u,35*v,"saturated liquid(6)"); outtextxy(55*u,34*v,"Vapor(5)");
        outtextxy(69*u,14*v,"TURBINE"); outtextxy(69*u,15*v+4,"LOW PRESSURE");
        outtextxy(58*u,16*v,"TURBINE"); outtextxy(58*u,17*v+4,"HIGH PRESSURE");
        break;
    default: break;
    }
getch(); closegraph();
}
/*=============================================*/
void diagram(int num)
{
extern int xl, yl;
int u, v;

abrirgraf();
u=(xl+1)/80; v=(yl+1)/58;          /*Después,y no antes de abrirgraf();*/
rectangle(2*u,2*v,78*u,57*v); rectangle(3*u,3*v,77*u,56*v);      /*marcos*/
setfillstyle(1,9); floodfill(1,1,15);
setfillstyle(1,14);floodfill(2*u+1,2*v+1,15);
setcolor(LIGHTBLUE); moveto(8*u,14*v); lineto(8*u,36*v); lineto(29*u,36*v);/*ejes*/
moveto(31*u,14*v); lineto(31*u,36*v);lineto(52*u,36*v);
moveto(54*u,14*v); lineto(54*u,36*v); lineto(76*u,36*v);
settextstyle(SMALL_FONT,HORIZ_DIR,6);
outtextxy(8*u,11*v,"P"); outtextxy(31*u,11*v,"logP"); outtextxy(53*u,11*v,"T");
outtextxy(29*u,37*v,"v"); outtextxy(52*u,37*v,"h"); outtextxy(75*u,37*v,"S");
setcolor(WHITE); moveto(9*u,36*v) ;lineto(10*u,27*v); lineto(11*u,20*v); /*curvas*/
lineto(12*u,17*v); lineto(13*u,15*v); lineto(15*u,14*v); lineto(16*u,14*v);
lineto(18*u,15*v); lineto(20*u,17*v); lineto(29*u,33*v);
moveto(32*u,36*v); lineto(34*u,30*v); lineto(36*u,25*v); lineto(38*u,21*v);
lineto(40*u,18*v); lineto(42*u,16*v); lineto(44*u,15*v); lineto(46*u,15*v);
lineto(48*u,16*v); lineto(49*u,18*v); lineto(49*u,20*v); lineto(48*u,27*v); lineto(47*u,36*v);
moveto(54*u,35*v); lineto(56*u,34*v); lineto(58*u,32*v); lineto(60*u,28*v);
lineto(62*u,22*v); lineto(63*u,21*v); lineto(65*u,20*v); lineto(66*u,20*v);
lineto(68*u,21*v); lineto(69*u,22*v); lineto(72*u,28*v); lineto(75*u,33*v); lineto(77*u,34*v);
circle(10*u,17*v,0.5*u); circle(12*u,17*v,0.5*u); circle(20*u,17*v,0.5*u);
circle(10*u,27*v,0.5*u); circle(35*u,18*v,0.5*u); circle(40*u,18*v,0.5*u);
circle(49*u,18*v,0.5*u); circle(44*u,30*v,0.5*u); circle(34*u,30*v,0.5*u);
circle(58*u,29*v,0.5*u); circle(62*u,22*v,0.5*u); circle(69*u,22*v,0.5*u); circle(58*u,32*v,0.5*u);
setlinestyle(0,0,3); setcolor(14);
moveto(10*u,17*v); lineto(20*u,17*v); moveto(22*u,27*v);      /*lineas*/
lineto(10*u,27*v); lineto(10*u,17*v); moveto(35*u,18*v); lineto(49*u,18*v);
moveto(44*u,30*v); lineto(34*u,30*v); lineto(35*u,18*v); moveto(58*u,29*v); lineto(62*u,22*v);
lineto(69*u,22*v); moveto(69*u,32*v); lineto(58*u,32*v); lineto(58*u,29*v);
```

```
switch (num)
    {
    case 1: line(20*u,17*v,22*u,27*v); line(49*u,18*v,44*u,30*v); line(69*u,22*v,69*u,32*v);
        moveto(4*u,50*v); lineto(5*u,49*v); lineto(5*u,51*v);       /*1*/
        moveto(4*u,52*v); lineto(5*u,52*v); lineto(5*u,53*v);       /*2*/
        lineto(4*u,53*v); lineto(4*u,54*v); lineto(5*u,54*v);
        moveto(23*u,49*v); lineto(24*u,49*v); lineto(24*u,51*v);    /*3*/
        lineto(23*u,51*v);line(23*u,50*v,24*u,50*v);
        moveto(23*u,52*v); lineto(23*u,53*v); lineto(24*u,53*v); line(24*u,52*v,24*u,54*v); /*4*/
        moveto(43*u,49*v); lineto(42*u,49*v); lineto(42*u,50*v);    /*5*/
        lineto(43*u,50*v); lineto(43*u,51*v); lineto(42*u,51*v);
        settextstyle(SMALL_FONT,HORIZ_DIR,6);
        outtextxy(23*u,5*v,"IDEAL AND SIMPLE RANKINE CYCLE");
        outtextxy(25*u,52*v,"Vapor");
        outtextxy(6*u,49*v,"Liquid");outtextxy(6*u,52*v,"satur.liquid");
        outtextxy(25*u,49*v,"satur.vapor");outtextxy(44*u,49*v,"satur.liquid");
        setcolor(15); settextstyle(SMALL_FONT,HORIZ_DIR,5);  /*Titulos*/
        outtextxy(3.5*u,40*v,"Ta=T2=T3=saturation temperature  P1=P2=P3=saturation
                                                    pressure  s3=s4");
        outtextxy(3.5*u,44*v,"Tb=T4=T5=condensation temperature P4=P5=condensation
                                                    pressure  s5=s1");
        copimag(4*u,49*v,5*u,51*v,9*u,14*v,0);copimag(4*u,49*v,5*u,51*v,34*u,15*v,0);
        copimag(4*u,49*v,5*u,51*v,57*u,26*v,0);copimag(4*u,52*v,5*u,54*v,12*u,18*v,0);
        copimag(4*u,52*v,5*u,54*v,39*u,15*v,0);copimag(4*u,52*v,5*u,54*v,61*u,19*v,0);
        copimag(23*u,49*v,24*u,51*v,21*u,15*v,0);copimag(23*u,49*v,24*u,51*v,49*u,15*v,0);
        copimag(23*u,49*v,24*u,51*v,70*u,20*v,0);copimag(23*u,52*v,24*u,54*v,21*u,28*v,0);
        copimag(23*u,52*v,24*u,54*v,43*u,31*v,0);copimag(23*u,52*v,24*u,54*v,68*u,33*v,0);
        copimag(42*u,49*v,43*u,51*v,10*u,28*v,0);copimag(42*u,49*v,43*u,51*v,34*u,31*v,0);
        copimag(42*u,49*v,43*u,51*v,58*u,33*v,0);
        circle(69*u,32*v,0.5*u);circle(22*u,27*v,0.5*u);
        break;
    case 2: line(20*u,17*v,22*u,27*v); line(49*u,18*v,44*u,30*v); line(69*u,22*v,69*u,32*v);
        setlinestyle(DOTTED_LINE,NORM_WIDTH,0);
        line(20*u,17*v,24*u,27*v);line(49*u,18*v,46*u,30*v);line(69*u,22*v,71*u,32*v);
        setlinestyle(SOLID_LINE,NORM_WIDTH,0);
        line(22*u,27*v,24*u,27*v);line(44*u,30*v,46*u,30*v);line(69*u,32*v,71*u,32*v);
        circle(24*u,27*v,0.5*u);circle(46*u,30*v,0.5*u);circle(71*u,32*v,0.5*u);
        setcolor(14);settextstyle(SMALL_FONT,HORIZ_DIR,6);
        outtextxy(23*u,5*v,"REAL AND SIMPLE RANKINE CYCLE");
        settextstyle(SMALL_FONT,HORIZ_DIR,5);
        outtextxy(5*u,40*v,"1s =Liquid (ideal)"); outtextxy(5*u,42*v,"1a =Liquid (real)");
        outtextxy(5*u,44*v,"2  =saturated liquid"); outtextxy(5*u,46*v,"3  =saturated vapor");
        outtextxy(5*u,48*v,"4s =Vapor (ideal)"); outtextxy(5*u,50*v,"4a =Vapor (real)");
        outtextxy(5*u,52*v,"5  =saturated vapor");
        copimag(5*u,40*v,7*u,42*v,9*u,14*v,0);copimag(5*u,40*v,7*u,42*v,34*u,15*v,0);
        copimag(5*u,40*v,7*u,42*v,56*u,26*v,0);copimag(5*u,44*v,7*u,46*v,12*u,18*v,0);
        copimag(5*u,44*v,7*u,46*v,39*u,15*v,0);copimag(5*u,44*v,7*u,46*v,61*u,19*v,0);
        copimag(5*u,46*v,7*u,48*v,21*u,15*v,0);copimag(5*u,46*v,7*u,48*v,49*u,15*v,0);
        copimag(5*u,46*v,7*u,48*v,70*u,20*v,0);copimag(5*u,48*v,7*u,50*v,21*u,28*v,0);
        copimag(5*u,48*v,7*u,50*v,43*u,31*v,0);copimag(5*u,48*v,7*u,50*v,68*u,33*v,0);
        copimag(5*u,50*v,7*u,52*v,24*u,28*v,0);copimag(5*u,50*v,7*u,52*v,46*u,31*v,0);
        copimag(5*u,50*v,7*u,52*v,71*u,33*v,0);copimag(5*u,52*v,7*u,54*v,10*u,28*v,0);
        copimag(5*u,52*v,7*u,54*v,34*u,31*v,0);copimag(5*u,52*v,7*u,54*v,58*u,33*v,0);
        setcolor(15);outtextxy(30*u,40*v,"Ta=T2=T3");
        outtextxy(30*u,42*v,"Tb=T5=T4s=T4a"); outtextxy(30*u,44*v,"x3=1  x2=x5=0");
        outtextxy(30*u,46*v,"P1s=P1a=P2=P3");outtextxy(30*u,48*v,"P5=P4s=P4a");
        outtextxy(30*u,50*v,"s4s=s3  s1s=s5"); circle(69*u,32*v,0.5*u); circle(22*u,27*v,0.5*u);
        break;
    case 3: moveto(69*u,22*v); lineto(70*u,18*v); lineto(70*u,32*v); lineto(69*u,32*v);
        moveto(49*u,18*v); lineto(51*u,18*v); lineto(44*u,30*v);
        moveto(20*u,17*v); lineto(21*u,17*v); lineto(22*u,27*v);
        setlinestyle(DOTTED_LINE,NORM_WIDTH,0);
        line(21*u,17*v,24*u,27*v);line(51*u,18*v,46*u,30*v);line(70*u,18*v,72*u,32*v);
        line(24*u,27*v,22*u,27*v);line(46*u,30*v,44*u,30*v);line(72*u,32*v,70*u,32*v);
        setlinestyle(SOLID_LINE,NORM_WIDTH,0);settextstyle(SMALL_FONT,HORIZ_DIR,6);
        outtextxy(23*u,5*v,"SUPERHEATED RANKINE CYCLE");
        settextstyle(SMALL_FONT,HORIZ_DIR,5);
        outtextxy(5*u,40*v,"1s =Liquid (ideal)"); outtextxy(5*u,42*v,"1a =Liquid (real)");
        outtextxy(5*u,44*v,"2  =saturated liquid"); outtextxy(5*u,46*v,"3  =saturated vapor");
        outtextxy(5*u,48*v,"4  =superheated vapor"); outtextxy(5*u,50*v,"5s =Vapor (ideal)");
        outtextxy(5*u,52*v,"5a =Vapor (real)"); outtextxy(5*u,54*v,"6  =saturated liquid");
        copimag(5*u,40*v,7*u,42*v,9*u,14*v,0);copimag(5*u,40*v,7*u,42*v,34*u,15*v,0);
        copimag(5*u,40*v,7*u,42*v,56*u,26*v,0);copimag(5*u,44*v,7*u,46*v,12*u,18*v,0);
        copimag(5*u,44*v,7*u,46*v,39*u,15*v,0);copimag(5*u,44*v,7*u,46*v,61*u,19*v,0);
        copimag(5*u,46*v,7*u,48*v,18*u,18*v,0);copimag(5*u,46*v,7*u,48*v,49*u,15*v,0);
        copimag(5*u,46*v,7*u,48*v,67*u,23*v,0);copimag(5*u,48*v,7*u,50*v,21*u,15*v,0);
        copimag(5*u,48*v,7*u,50*v,51*u,15*v,0);copimag(5*u,48*v,7*u,50*v,70*u,15*v,0);
        copimag(5*u,50*v,7*u,52*v,22*u,28*v,0);copimag(5*u,50*v,7*u,52*v,43*u,31*v,0);
        copimag(5*u,50*v,7*u,52*v,68*u,33*v,0);copimag(5*u,52*v,7*u,54*v,24*u,28*v,0);
        copimag(5*u,52*v,7*u,54*v,46*u,31*v,0);copimag(5*u,52*v,7*u,54*v,73*u,33*v,0);
        copimag(5*u,54*v,7*u,55.5*v,10*u,28*v,0);copimag(5*u,54*v,7*u,55.5*v,34*u,31*v,0);
        copimag(5*u,54*v,7*u,55.5*v,58*u,33*v,0);setcolor(15);
        outtextxy(30*u,40*v,"Ta=T2=T3");outtextxy(30*u,42*v,"Tsh=T4");
        outtextxy(30*u,44*v,"Tb=T5s=T5a=T6"); outtextxy(30*u,46*v,"x3=1  x2=x6=0");
        outtextxy(30*u,48*v,"P1s=P1a=P2=P3=P4");outtextxy(30*u,50*v,"P6=P5s=P5a");
        outtextxy(30*u,52*v,"s4=s5s  s1s=s6");
        circle(21*u,17*v,0.5*u);circle(22*u,27*v,0.5*u);circle(24*u,27*v,0.5*u);
        circle(51*u,18*v,0.5*u);circle(46*u,30*v,0.5*u);circle(70*u,18*v,0.5*u);
```

```
    circle(70*u,32*v,0.5*u);circle(72*u,32*v,0.5*u);
    break;
case 4: moveto(69*u,22*v); lineto(70*u,18*v); lineto(70*u,23*v); lineto(73*u,18*v)
    lineto(73*u,32*v); lineto(69*u,32*v); moveto(49*u,18*v); lineto(51*u,18*v);
    lineto(49*u,22*v); lineto(53*u,22*v); lineto(44*u,30*v); moveto(20*u,17*v);
    lineto(21*u,17*v); lineto(22*u,20*v); lineto(23*u,20*v); lineto(24*u,27*v);
    lineto(22*u,27*v);setlinestyle(DOTTED_LINE,NORM_WIDTH,0);
    line(23*u,20*v,25*u,27*v);line(51*u,18*v,51*u,22*v);line(53*u,22*v,46*u,30*v);
    line(70*u,18*v,71*u,21*v);line(73*u,18*v,74*u,32*v);line(46*u,30*v,44*u,30*v);
    setlinestyle(SOLID_LINE,NORM_WIDTH,0);settextstyle(SMALL_FONT,HORIZ_DIR,6);
    outtextxy(23*u,5*v,"REHEATED RANKINE CYCLE");
    settextstyle(SMALL_FONT,HORIZ_DIR,5);
    outtextxy(5*u,40*v,"1s =Liquid (ideal)"); outtextxy(5*u,42*v,"1a =Liquid (real)");
    outtextxy(5*u,44*v,"2  =saturated liquid"); outtextxy(5*u,46*v,"3  =saturated vapor");
    outtextxy(5*u,48*v,"4  =superheated vapor(high press.)");
    outtextxy(5*u,50*v,"5s =superheat.vapor(low press.)(ideal)");
    outtextxy(5*u,52*v,"5a =superheat.vapor(low press.)(real)");
    outtextxy(27*u,40*v,"5' =superheat.vapor (low press.) reheated");
    outtextxy(27*u,42*v,"5''s =Vapor (ideal)");
    outtextxy(27*u,44*v,"5''a =Vapor (real)"); outtextxy(27*u,46*v,"6 =saturated liquid");
    copimag(5*u,40*v,7*u,42*v,9*u,14*v,0);copimag(5*u,40*v,7*u,42*v,34*u,15*v,0);
    copimag(5*u,40*v,7*u,42*v,56*u,26*v,0);copimag(5*u,44*v,7*u,46*v,12*u,18*v,0);c
    copimag(5*u,44*v,7*u,46*v,39*u,15*v,0);copimag(5*u,44*v,7*u,46*v,61*u,19*v,0);
    copimag(5*u,46*v,7*u,48*v,18*u,18*v,0);copimag(5*u,46*v,7*u,48*v,49*u,15*v,0);
    copimag(5*u,46*v,7*u,48*v,67*u,23*v,0);copimag(5*u,48*v,7*u,50*v,21*u,15*v,0);
    copimag(5*u,48*v,7*u,50*v,51*u,15*v,0);copimag(5*u,48*v,7*u,50*v,70*u,15*v,0);
    copimag(5*u,50*v,7*u,52*v,20*u,20*v,0);copimag(5*u,50*v,7*u,52*v,47*u,22*v,0);
    copimag(5*u,50*v,7*u,52*v,70*u,24*v,0);copimag(5*u,52*v,7*u,54*v,50*u,23*v,0);
    copimag(5*u,52*v,7*u,54*v,71*u,22*v,0);copimag(27*u,40*v,29*u,42*v,24*u,18*v,0);
    copimag(27*u,40*v,29*u,42*v,53*u,19*v,0);copimag(27*u,40*v,29*u,42*v,73*u,15*v,0);
    copimag(27*u,42*v,30*u,44*v,20*u,28*v,0);copimag(27*u,42*v,30*u,44*v,42*u,31*v,0);
    copimag(27*u,42*v,30*u,44*v,69*u,29*v,0);copimag(27*u,44*v,30*u,46*v,25*u,28*v,0);
    copimag(27*u,44*v,30*u,46*v,47*u,30*v,0);copimag(27*u,44*v,30*u,46*v,73*u,33*v,0);
    copimag(27*u,46*v,28*u,48*v,10*u,28*v,0);copimag(27*u,46*v,28*u,48*v,34*u,31*v,0);
    copimag(27*u,46*v,28*u,48*v,58*u,33*v,0);
    setcolor(15); outtextxy(50*u,42*v,"Ta=T2=T3  Tsh=T4=T5'");
    outtextxy(50*u,44*v,"Tb=T5''s=T5''a=T6"); outtextxy(50*u,46*v,"x3=1  x2=x6=0");
    outtextxy(50*u,48*v,"P1s=P1a=P2=P3=P4");outtextxy(50*u,50*v,"P6=P5''s=P5''a");
    outtextxy(50*u,52*v,"P5s=P5a=P5'   s4=s5s s1=s6");
    circle(21*u,17*v,0.5*u);circle(22*u,20*v,0.5*u);circle(23*u,20*v,0.5*u);
    circle(24*u,27*v,0.5*u);circle(25*u,27*v,0.5*u);circle(51*u,18*v,0.5*u);
    circle(49*u,22*v,0.5*u);circle(51*u,22*v,0.5*u);circle(53*u,22*v,0.5*u);
    circle(46*u,30*v,0.5*u);circle(70*u,18*v,0.5*u);circle(70*u,23*v,0.5*u);
    circle(73*u,18*v,0.5*u);circle(71*u,21*v,0.5*u);circle(73*u,32*v,0.5*u);
        circle(74*u,32*v,0.5*u); break;
    default: break;
    }
getch(); closegraph();
}
/*=========================================*/
void exp_rs()
{
char tecla;
do  {
    textrankrs(); tecla=toupper(getch());
    switch(tecla)
        {
        case 'S': esquema(1); break;
        case 'D': diagram(2); break;
        case 'X': break;
        default: puts("\a");
        }
    }
    while(tecla!='X');
}
/*=========================================*/
void textrankrs(void)
{
clrscr(); textmode(C80); textcolor(15); textbackground(9);
cputs("                  REAL AND SIMPLE RANKINE CYCLE                  ");
cputs("                                                                 ");
cputs("   Real and simple Rankine cycle is the real case. Now:          ");
cputs("-- There are irreversibilities. The most importants are in the pump and turbine.");
cputs("-- Then,process (3-4a) and (5-1a) are irreveribles,so s3 is not equal to s4a. ");
cputs("                  (a) = real case                                ");
cputs("                  (s) = ideal case                               ");
cputs("   Now, entalpy of (3 and 4a) and (5 and 1a ) are not equal.     ");
cputs("                                                                 ");
cputs("For the efficiency calculus of the cycle,Ea, we must know the next efficiencies:");
cputs("                    W(3-4a)                  W(5-1s)   ");
cputs("     Et=turbine efficiency= -------     Ep=pump efficiency = -------  ");
cputs("                    W(3-4s)                  W(5-1a)   ");
cputs("                       W(3-4a)+W(5-1a)                   ");
cputs("  The cycle efficiency is:   Ea = - ----------------  Es = ideal efficiency ");
cputs("                              Q(1a-3)                    ");
cputs("                                                         ");
cputs("where W(3-4a)=h4a-h3 , W(5-1a)=h1a-h5 and  Q(1a-2)=h2-h1a          ");
cputs("Q(1a-2)Q(1s-2),then energy added to the cycle is greater than the real case. ");
```

```
    cputs("*********************************************************************************
");
    cputs("*  Options:    D == DIAGRAMS      S == SCHEME      X == EXIT       *");
    cputs("*********************************************************************************
");
    }
```

EXPLN2.C

```
#include "rank.h"
/*=========================================*/
void exp_sup()
{
char tecla;
do  {
    textranksup(); tecla=toupper(getch());
    switch(tecla)
        {
        case 'S': esquema(3); break;
        case 'D': diagram(3); break;
        case 'X': break;
        default: puts("\a");
        }
    } while(tecla!='X');
}
/*=========================================*/
void exp_rec()
{
char tecla;
do  {
    textrankrec(); tecla=toupper(getch());
    switch(tecla)
        {
        case 'S': esquema(4); break;
        case 'D': diagram(4); break;
        case 'X': break;
        default: puts("\a");
        }
    } while(tecla!='X');
}
/*=========================================*/
void textranksup(void)
{
clrscr(); textmode(C80); textcolor(15); textbackground(9);
cputs("              SUPERHEATED RANKINE CYCLE                    ");
cputs("  This cycle is different to the previous one because the boiler output is");
cputs("connected to a superheater, which has got a pressure equal to the saturated");
cputs("pressure of the boiler, and it is heated until the temperature Tsh. Then, the");
cputs("saturated vapor entering into the superheater increases its temperature,with the");
cputs("same pressure. At the output we have SUPERHEATED SATURATED VAPOR, state 4. The");
cputs("cycle continues equal to the simple case.                  ");
cputs("                                                           ");
cputs("  Efficiency calculus:                                     ");
cputs("                    W(4-5)+W(6-1)                          ");
cputs("  Efficiency: Esup= - ----------------------------------   ");
cputs("                   |Q(1-2)+Q(2-3)+Q(3-4)|                  ");
cputs("  W(4-5s)=h[5s]-h[4]              W(4-5a)=h[5a]-h[4]       ");
cputs("  W(6-1s)=h[1s]-h[6]  Q(2-3)=h[3]-h[2]  W(6-1a)=h[1a]-h[6] ");
cputs("  Q(1s-2)=h[2]-h[1s]  Q(3-4)=h[4]-h[3]  Q(1a-2)=h[2]-h[1a] ");
cputs("                                                           ");
cputs("  Obviously, Esup,a  Esup,s. Esup,aEa because now we have a greater output");
cputs("work in the turbine ( W(4-5a) is greater than W(3-4) in the simple case ),");
cputs("because the saturated vapor entering into turbine (it is superheated) has got");
cputs("a higher entalpy. Also, quality (x) of the vapor leaving the turbine is higher. ");
cputs("*********************************************************************************
");
cputs("*  Options:    S == SCHEME      D == DIAGRAMS      X == EXIT      *");
cputs("*********************************************************************************
");
}
/*=========================================*/
void textrankrec(void)
{
clrscr();textmode(C80);textcolor(15);textbackground(9);
cputs("              REHEATED RANKINE CYCLE                       ");
cputs("  Reheated Rankine cycle has got new characteristics. The reheated saturated");
cputs("vapor (4) has got a high pressure at the reheater output. Then it goes through");
cputs("a turbine ( high pressure ); at the output, we have superheated vapor at low ");
cputs("lpressure (5), having a cooling and reducing its temperature and pressure,");
cputs("according with the last turbine. After, it enters again into the superheater,");
cputs("reaching the Tsh temperature but keeping low its pressure, so at the superheater");
cputs("output we have the reheated and with low pressure superheated vapor state (5').");
cputs("Then, it passes through a turbine (low pressure) at wich output we see the vapor");
```

```
    cputs("state ( 5''). The cycle continues from here in the same description of the");
    cputs("previous cycles.                                                          ");
    cputs("    This cycle efficiency is:                                             ");
    cputs("                                                                          ");
    cputs("              W(4-5)+W(5'-5'')+W(6-1)                                     ");
    cputs("    Ereh = - -----------------------------                                ");
    cputs("           IQ(1-2)+Q(2-3)+Q(3-4)+Q(5-5')I                                 ");
    cputs("                                                                          ");
    cputs("    Ereh Esup because with the two turbines we have increased the util power");
    cputs("of the cycle,and then it increases its efficiency.                        ");
    cputs("                                                                          ");
    cputs("********************************************************************************
");
    cputs("* Options:    S == SCHEME       D == DIAGRAMS       X == EXIT     *");
    cputs("********************************************************************************
");
    }
    /*=======================================*/
    void error_message(void)
    {
    textmode(C80);clrscr();textbackground(LIGHTBLUE);textcolor(YELLOW);
    cputs("    With these input datas you have introduced , this cycle is'nt available.");
    cputs("The reason is because the calculations give us a vapor real state which has");
    cputs("got an enthalpy greater than the corresponding to the saturated vapor state.");
    cputs("Then,the vapor real state is thermodynamically unreachable.               ");
    cputs("    The probable causes of this event are:                                ");
    cputs("    - Pump and/or turbine efficiencies very low.Pump efficiency can be very low");
    cputs("      but turbine efficiency can not be it.                               ");
    cputs("    - Condensation temperature (Tb) very high                             ");
    cputs("    - When a superheater is present,there is a relation between Ta , Tb and Tsh");
    cputs("      which can generate problems because the states go out the cycle limits. ");
    cputs("      Thus,we are going to have got a problem if Tsh is high when Tb is high or");
    cputs("      Tb is low.Here is the practical solution:                           ");
    cputs("          -When Ta is low:  down Tb  and/or  down Tsh                     ");
    cputs("          -When Tb is high: up   Ta  and/or  down Tsh                     ");
    cputs("    - When there is a reheat in the cycle,if we increase Tsh ,then the Ph range");
    cputs("      must be smaller:up Ph low limit and down Ph high limit.For example:    ");
    cputs("          - Tsh=300 : 500hê                                               ");
    cputs("          - Tsh=600 : 700hö                                               ");
    cputs("                                                                          ");
    cputs("Please,use the reference values in the \"input data\" window.              ");
    getch();
    }
```

RANK.PRJ

```
rankgraf.c      (rank.h)
rankis.c        (rank.h)
rankcalc.c      (rank.h)
rankrs.c        (rank.h)
expln1.c        (rank.h)
expln2.c        (rank.h)
att.obj
cga.obj
egavga.obj
herc.obj
goth.obj
litt.obj
trip.obj
sans.obj
```

Apéndice III:

LISTADO DEL PROGRAMA PLANTA

Apéndice III:

LISTADO DEL PROGRAMA PLANTA

Características:

Desarrollado en C++.

Corre en estaciones de trabajo (p.ej. SUN Sparcstation), compilando mediante GNU C++.

Utiliza X-Windows.

Se dedica a una simulación dinámica orientada a objeto.

PROGRAMA PLANTA

Makefile

```
CC = g++

#   CC = gcc
#   Por defecto, si no especificamos CC, tomara cc.
#   Codigo preparado para cc y gcc, pero no para gcc.
EXEC= planta
CFLAGS= -O
LIBS= -lX11 -lm
OBJECTS= initx.o          \
         quitx.o          \
         datos.o          \
         tablas.o         \
         funcions.o       \
         argsx.o          \
         textx.o          \
         cursores.o       \
         teclado.o        \
         windowx.o        \
         drawx.o          \
         eventos.o        \
         raton.o          \
         dibujos.o        \
         ayudas.o         \
         colorx.o         \
         menus.o          \
         clases.o         \
         mostrar.o        \
         esquema1.o       \
         esquema2.o       \
         esquema3.o       \
         main.o

$(EXEC): $(OBJECTS)
        g++     -o $(EXEC) $(OBJECTS) $(LIBS)

initx.c      : windows.h
quitx.c      : windows.h
argsx.c      : windows.h
colorx.c     : windows.h
drawx.c      : windows.h
textx.c      : windows.h
teclado.c    : windows.h
raton.c      : windows.h
dibujos.c    : windows.h
ayudas.c     : windows.h
eventos.c    : windows.h
windowx.c    : windows.h
menus.c      : windows.h
main.c       : windows.h
datos.c      :
tablas.c     :
funcions.c   : clases.h
clases.c     : clases.h
mostrar.c    : windows.h
esquema1.c   : windows.h     clases.h
esquema2.c   : windows.h     clases.h
esquema3.c   : windows.h     clases.h
```

clases.h

```
/* clases.h */
/*=============================================*/
extern int orden_interp_sat;
extern int orden_interp_vap;
extern int orden_interp_liq;
extern double a_sat[4], b_sat[4], c_sat[4], d_sat[4];
extern double T_sat[76], P_sat[76], h_sat_v[76], s_sat_v[76], v_sat_v[76];
extern double h_sat_l[76], s_sat_l[76], v_sat_l[76];
extern double c_vap[4][11], a_vap[4][11];
extern double x_dat[11];
extern double P_sup[28], TK_sup[28][18], h_sup[28][18];
extern double  v_sup[28][18], s_sup[28][18];
extern double P_l[27], T_l[27][22], h_l[27][22], v_l[27][22];
extern double e_liq[4][22];
extern double T_dat[27];
extern double *arrT_l, *arrh_l, *arrv_l;
extern double *arrTK_sup, *arrh_sup, *arrv_sup, *arrs_sup;
/*---------------------------------------------*/
/* funcions.c */
double intrplcn(double x1, double x2, double y1, double y2, double x0);
double tabla_1d(double x0, double y[], double x[], int size);
double tabla_2d_xy(double x[], int xsize, double y[], int ysize, double z[], double x0, double y0);
double tabla_2d_xz(double x[], int xsize, double y[], int ysize, double z[], double x0, double z0);
double tabla_2d_xz_sh(double x[], int xsize, double y[], int ysize, double z[], double x0, double z0);
double tabla_2d_xy_sh(double x[], int xsize, double y[], int ysize, double z[], double x0, double y0);
double tabla_2d_xy_liq(double x[], int xsize, double y[], int ysize, double z[], double x0, double y0);
double tabla_2d_xz_liq(double x[], int xsize, double y[], int ysize, double z[], double x0, double z0);
/*=============================================*/
class Liquido_saturado
{
protected:
    double densidad, temperatura, entalpia, entropia, presion;
    double masa, Volumen, volumen_especifico;
public:
    Liquido_saturado() { }
    void set_pyV(double p, double V);
    double dar_densidad(void) { return densidad; }
    double dar_entalpia(void) { return entalpia; }
    double dar_Volumen(void) { return Volumen; }
    double dar_entropia(void) { return entropia; }
    double dar_presion(void) { return entropia; }
    double dar_volumen_especifico(void) { return volumen_especifico; }
    double dar_temperatura(void) { return temperatura; }
    double dar_masa(void) { return masa; }
};
/*---------------------------------------------*/
class Vapor_saturado
{
protected:
    double densidad, temperatura, entalpia, entropia, presion;
    double masa, Volumen, volumen_especifico;
public:
    Vapor_saturado() { }
    void set_pyV(double p, double V);
    double dar_densidad(void) { return densidad; }
    double dar_entalpia(void) { return entalpia; }
    double dar_Volumen(void) { return Volumen; }
    double dar_entropia(void) { return entropia; }
    double dar_presion(void) { return entropia; }
    double dar_volumen_especifico(void) { return volumen_especifico; }
    double dar_temperatura(void) { return temperatura; }
    double dar_masa(void) { return masa; }
};
/*---------------------------------------------*/
class Vapor
{
protected:
    double presion, temperatura, densidad;    /* variables termodin<Nmicas*/
    double entalpia, entropia, calidad, volumen_especifico;
    int num_estado; /*num_estado:0=vapor1=vapor saturado2=supercalentado*/
    char *estado;
public:
    Vapor() { }
    void setPyh(double P, double h);
    void setPys(double P, double s);
    double dar_T(void) { return temperatura; }
    double dar_P(void) { return presion; }
    double dar_v(void) { return volumen_especifico; }
    double dar_s(void) { return entropia; }
    double dar_h(void) { return entalpia; }
    double dar_x(void) { return calidad; }
    double dar_d(void) { return densidad; }
    char *dar_estado(void) { return estado; }
};
/*---------------------------------------------*/
```

```
class Valvula
{
protected:
    double W;
public:
    Valvula() { }
    Valvula(double flujo) { W=flujo; }
    double dar_W(void) { return W; }
    void mas_W(void) { W=W+0.001; }
    void menos_W(void) { if(W>=0.001) W=W-0.001; }
};
/*-----------------------------------------------------------------*/
class Fuente_de_liquido
{
protected:
    double h;
public:
    Fuente_de_liquido() { }
    Fuente_de_liquido(double entalpia) { h=entalpia; }
    double dar_h(void) { return h; }
    void mas_h(void) { h=h+10.; }
    void menos_h(void) { if(h>=10.) h=h-10.; }
};
/*-----------------------------------------------------------------*/
class Quemador
{
protected:
    double Q;
public:
    Quemador(double calor) { Q=calor; }
    double dar_Q(void) { return Q; }
    void set_Q(double val) { Q=val; }
    void mas_Q(void)   { Q=Q+1.; }
    void menos_Q(void) { if(Q>=1) Q=Q-1.; }
};
/*-----------------------------------------------------------------*/
class Caldera
{
protected:
    double Wlec, Wvsc, Lc, nc, Rc, Pc, Qec, hlec;
    double A0, A1, A2, A3, A4, k1, k2, k3, k4, A, B;
    double dvc, dlc, hvc, hlc, vvc, vlc, mvc, mlc, slc, svc, mc, tc;
    Liquido_saturado liqcal;
    Vapor_saturado vapcal;
public:
    Caldera() { }
    Caldera(double radio, double longitud, double nivel, double presion)
        { Rc=radio; Lc=longitud; nc=nivel; Pc=presion; }
    void calculos_t(void);
    void inicializo_ncyPc(double h);
    double dar_Qec(void) { return Qec; }
    double dar_Wlec(void) { return Wlec; }
    double dar_Wvsc(void) { return Wvsc; }
    double dar_dlc(void) { return dlc; }
    double dar_dvc(void) { return dvc; }
    double dar_hlc(void) { return hlc; }
    double dar_hvc(void) { return hvc; }
    double dar_mlc(void) { return mlc; }
    double dar_mvc(void) { return mvc; }
    double dar_slc(void) { return slc; }
    double dar_svc(void) { return svc; }
    double dar_nc(void) { return nc; }
    double dar_Pc(void) { return Pc; }
    double dar_Lc(void) { return Lc; }
    double dar_tc(void) { return tc; }
    double dar_Rc(void) { return Rc; }
    void set_variables(double calor_in, double flujo_in, double entalpia_in, double flujo_out)
        { Qec=calor_in; Wlec=flujo_in; hlec=entalpia_in; Wvsc=flujo_out; }
};
/*-----------------------------------------------------------------*/
class Supercalentador
{
protected:
    double Tves, Tvss, Pves, Pvss, hves, hvss, vves, vvss, sves, svss;
    double Qes, Wves, Wvss, mvs, k, ksup;
    Vapor vapor_salida;
    char *estado_salida;

public:
    Supercalentador() { }
    Supercalentador(double h, double m, double k)
        { hvss=h; mvs=m; ksup=k; }
    void set_variables(double P, double T, double h, double v, double Q, double W, double s)
        { Pves=P; Tves=T; hves=h; vves=v; Qes=Q; Wves=W; sves=s; }
    void calculos_t(double t, double W);
    void inicializo_hvss(double t);
    double dar_Wves(void) { return Wves; }
    double dar_Wvss(void) { return Wvss; }
```

```
double dar_Tves(void) { return Tves; }
double dar_Tvss(void) { return Tvss; }
double dar_Pves(void) { return Pves; }
double dar_Pvss(void) { return Pvss; }
double dar_hves(void) { return hves; }
double dar_hvss(void) { return hvss; }
double dar_vves(void) { return vves; }
double dar_vvss(void) { return vvss; }
double dar_sves(void) { return sves; }
double dar_svss(void) { return svss; }
double dar_mvs(void)  { return mvs; }
double dar_ksup(void) { return ksup; }
char*dar_estado_salida(void)
    { return estado_salida; }
};
/*-------------------------------------------------------------------*/
class Turbina
{
protected:
    double Wvet, Pvet, Tvet, hvet, svet, vvet;
    double Wvst, Pvst, Tvst, hvst, svst, vvst, xvst;
    double Wtx, kx, Ri, hvsti;
    void salida(void);
    Vapor vap_sal;
    Vapor vap_sal_i;
    char *estado_salida;
public:
    Turbina() { }
    Turbina(double k) { kx=k; }
    void set_variables(double W, double P, double T, double h, double s, double v, double P2 )
        { Wvet=W; Pvet=P; Tvet=T; hvet=h; svet=s; vvet=v; Pvst=P2; salida(); }
    char *dar_estado_salida(void) { return estado_salida; }
    double dar_hvst(void) { return hvst; }
    double dar_Tvst(void) { return Tvst; }
    double dar_vvst(void) { return vvst; }
    double dar_xvst(void) { return xvst; }
    double dar_Wvst(void) { return Wvst; }
    double dar_kx(void)   { return kx; }
    void mas_kx(void)
        { if(kx<1.0) kx=kx+0.1; }<M=
    void menos_kx(void)
        { if(kx>=0.1) kx=kx-0.1; }
};
/*-------------------------------------------------------------------*/
```



```
class Bomba
{
protected:
    double Wb, hlsb, Pleb, Plsb, dleb, dlsb, Tlsb, vlsb;
    void salida(void)
        {
        dlsb=dleb; vlsb=1./dlsb;
        Tlsb=tabla_2d_xy_liq(P_l, 27, arrv_l, 22, arrT_l, Plsb, vlsb);
        hlsb=tabla_2d_xy_liq(P_l, 27, arrv_l, 22, arrh_l, Plsb, vlsb);
        }
public:
    Bomba() { }
    Bomba(double W) { Wb=W; }
    double dar_Wb(void) { return Wb; }
    double dar_hlsb(void) { return hlsb; }
    void set_variables(double P_in, double d_in, double P_out)
        { Pleb=P_in; dleb=d_in; Plsb=P_out; salida(); }
    double dar_Tlsb(void)  { return Tlsb; }
    void mas_Wb(void)
        { Wb=Wb+0.001; }
    void menos_Wb(void)
        { if(Wb>=0.001) Wb=Wb-0.001; }
};
/*-------------------------------------------------------------------*/
class Condensador
{
protected:
    double Rcd, Lcd, ncd, Pcd, Tcd, Wlscd;          /* datos varios */
    double Wvecd, hvecd, dvecd, xvecd, Tvecd, mvecd;    /*vapor de entrada*/
    double Vvcd, dm, hm, xm, mm,Tm;                 /* vapor intermedio */
    double Vlcd, dlcd, hlcd, mlcd;                  /* liquido saturado */
    double Wr, Tre, Trs, mr, dre, drs, hre, hrs;    /* refrigerante */
    double A0, A1, A2, A3, A4, A4p, A, B;
    double k11, k12, k2, k31, k32, k4, k5;
    double c_x[4], a_x[4], e_T[4];
    Liquido_saturado liq_cond;
    double hl, hg, vl, vg;
public:
    Condensador() { }
    Condensador(double radio, double longitud, double nivel, double presion,
            double Wref, double mref, double Tref_in, double Tref_out)
        { Rcd=radio; Lcd=longitud; ncd=nivel; Pcd=presion; Wr=Wref;
                                    mr=mref; Tre=Tref_in; Trs=Tref_out; }
    void set1_variables(void);
```

```
    void set2_variables(double Win, double T, double x, double h, double v, double Wout);
    double dar_Pcd(void)  { return Pcd; }
    double dar_dlcd(void) { return dlcd; }
    double dar_hlcd(void) { return hlcd; }
    double dar_mr(void)   { return mr; }
    double dar_Lcd(void)  { return Lcd; }
    double dar_ncd(void)  { return ncd; }
    double dar_Tcd(void)  { return Tcd; }
    double dar_Tre(void)  { return Tre; }
    double dar_Trs(void)  { return Trs; }
    double dar_Rcd(void)  { return Rcd; }
    double dar_Wr(void)   { return Wr; }
    void inicializo_ncdyPcd(double h);
    void mas_Wr(void) { Wr=Wr+0.001; }
    void menos_Wr(void) { if(Wr>=0.001) Wr=Wr-0.001; }
    void mas_Tre(void) { Tre=Tre+1.; }
    void menos_Tre(void) { if(Tre>=0.) Tre=Tre-1.; }
    void mas_Trs(void) { Trs=Trs+1.; }
    void menos_Trs(void) { if(Trs>=0.) Trs=Trs-1.; }
};
```


thelcon.h

```
/* thelcon.h */

#define thelcon_width 64
#define thelcon_height 64

static char thelcon_bits[] = {
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0xfc, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x1f,
        0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x04, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x30, 0xe4, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x73,
        0xe4, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x73, 0xe4, 0x03, 0x00, 0x00,
        0x00, 0x00, 0xe0, 0x73, 0xc4, 0x07, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x71,
        0x84, 0x0f, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x70, 0x04, 0x1f, 0x00, 0x00,
        0x00, 0x00, 0x7c, 0x7c, 0x04, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x3e, 0x70,
        0x04, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x70, 0x04, 0xf8, 0x00, 0x00,
        0x00, 0x80, 0x0f, 0x70, 0x04, 0xf0, 0x01, 0x00, 0x00, 0xc0, 0x07, 0x07,
```



```
0x04, 0xe0, 0x03, 0x00, 0x00, 0xe0, 0x03, 0x70, 0x04, 0xc0, 0x07, 0x00,
0x00, 0xf0, 0x01, 0x70, 0x04, 0x80, 0x0f, 0x00, 0x00, 0xf8, 0x00, 0x70,
0x04, 0x00, 0x1f, 0x00, 0x00, 0x7c, 0x00, 0x70, 0x04, 0x00, 0x3e, 0x00,
0x00, 0x3e, 0x00, 0x70, 0x04, 0x00, 0x7c, 0x00, 0x00, 0x1f, 0x00, 0x70,
0x04, 0x00, 0xf8, 0x00, 0x80, 0x0f, 0x00, 0x70, 0x04, 0x00, 0xf0, 0x01,
0xc0, 0x07, 0x00, 0x70, 0x04, 0x00, 0xe0, 0x03, 0xe0, 0x03, 0x00, 0x70,
0x04, 0x00, 0xc0, 0x07, 0xf0, 0x01, 0x00, 0x70, 0x04, 0x00, 0x80, 0x0f,
0xf8, 0x00, 0x00, 0x70, 0x04, 0x00, 0x00, 0x1f, 0x7c, 0x00, 0x00, 0x70,
0x04, 0x00, 0x00, 0x3e, 0x3e, 0x00, 0x00, 0x70, 0x04, 0x00, 0x00, 0x7c,
0x1f, 0x00, 0x00, 0x70, 0x04, 0x00, 0x00, 0xf8, 0x0f, 0x00, 0x00, 0x70,
0x04, 0x00, 0x00, 0xf0, 0x07, 0x00, 0x00, 0x70, 0x04, 0x00, 0x00, 0xf0,
0x03, 0x00, 0x00, 0x70, 0x04, 0x00, 0x00, 0xf0, 0x07, 0x00, 0x00, 0x70,
0x04, 0x00, 0x00, 0xf8, 0x0f, 0x00, 0x00, 0x70, 0x04, 0x00, 0x00, 0x7c,
0x1f, 0x00, 0x00, 0x70, 0x04, 0x00, 0x00, 0x3e, 0x3e, 0x00, 0x00, 0x70,
0x04, 0x00, 0x00, 0x1f, 0x7c, 0x00, 0x00, 0x70, 0x04, 0x00, 0x80, 0x0f,
0xf8, 0x00, 0x00, 0x70, 0x04, 0x00, 0xc0, 0x07, 0xf0, 0x01, 0x00, 0x70,
0x04, 0x00, 0xe0, 0x03, 0xe0, 0x03, 0x00, 0x70, 0x04, 0x00, 0xf0, 0x01,
0xc0, 0x07, 0x00, 0x70, 0x04, 0x00, 0xf8, 0x00, 0x80, 0x0f, 0x00, 0x70,
0x04, 0x00, 0x7c, 0x00, 0x00, 0x1f, 0x00, 0x70, 0x04, 0x00, 0x3e, 0x00,
0x00, 0x3e, 0x00, 0x70, 0x04, 0x00, 0x1f, 0x00, 0x00, 0x7c, 0x00, 0x70,
0x04, 0x80, 0x0f, 0x00, 0x00, 0xf8, 0x00, 0x70, 0x04, 0xc0, 0x07, 0x00,
0x00, 0xf0, 0x01, 0x70, 0x04, 0xe0, 0x03, 0x00, 0x00, 0xe0, 0x03, 0x70,
0x04, 0xf0, 0x01, 0x00, 0x00, 0xc0, 0x07, 0x70, 0x04, 0xf8, 0x00, 0x00,
0x00, 0x80, 0x0f, 0x70, 0x04, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x70,
0x04, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x3e, 0x70, 0x04, 0x1f, 0x00, 0x00,
0x00, 0x00, 0x7c, 0x70, 0x84, 0x0f, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x70,
0xc4, 0x07, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x71, 0xe4, 0x03, 0x00, 0x00,
0x00, 0x00, 0xe0, 0x73, 0xe4, 0x01, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x73,
0xe4, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x73, 0x04, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x70, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70,
0xfc, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x7f, 0xf0, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0x7f, 0xe0, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x7f,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

windows.h

```
/* windows.h */

#define EV_MASK ( ButtonPressMask | KeyPressMask | ExposureMask |StructureNotifyMask )

#define VENTANA_NORMAL 0
#define VENTANA_POP_UP 1

#define ESTADO_NORMAL 0
#define ESTADO_ICONO 1

#define RES_NAME     "central"   /* Resource name */
#define RES_CLASS    "clase"   /* Resource Class, e.g. "editor" */
/*=============================================*/
/* initx.c */
void initX(char *nombre_display);
void info_X(void);
/*---------------------------------------------------------------*/
/* quitx.c */
void quitX(void);
void limpia_ventanas_GCs(void);
/*---------------------------------------------------------------*/
/* argsx.c */
void print_mensaje_ayuda(void);
int get_argumentos(int argc,char *argv[], char nombre_display[], char geometria[],
                   char nombre_font[], char titulo[]);
/*---------------------------------------------------------------*/
/* colorx.c */
void set_color_nombre(GC el_GC, char theName[]);
void init_colores_defecto(void);
void set_color(GC el_GC, int colorNumber);
/*---------------------------------------------------------------*/
/* drawx.c */
void dibujaLinea(Window ventana, GC el_GC, int x1, int y1, int x2, int y2);
void dibujaPunto(Window ventana, GC el_GC, int x, int y);
void dibujaRectangulo(Window ventana, GC el_GC, int x, int y, int ancho, int alto);
void llenaRectangulo(Window ventana, GC el_GC, int x, int y, int ancho, int alto);
void dibujaOvalo(Window ventana,GC el_GC,int x,int y,int ancho,int alto);
void llenaOvalo(Window ventana, GC el_GC, int x, int y, int ancho, int alto);
void dibujaArco(Window ventana, GC el_GC, int x, int y, int ancho, int alto,
                int comienzo_angulo, int angulo);
void llenaArco(Window ventana, GC el_GC, int x, int y, int ancho, int alto,
                int comienzo_angulo, int angulo);
void dibujaPuntos(Window ventana, GC el_GC, XPoint *losPuntos, int elNumeroDePuntos, int elModo);
void dibujaLineas(Window ventana, GC el_GC, XPoint *losPuntos, int elNumeroDePuntos, int elModo);
void dibujaArcos(Window ventana, GC el_GC, XArc *losArcos, int elNumeroDeArcos);
void llenaArcos(Window ventana, GC el_GC, XArc *losArcos, int elNumeroDeArcos);
void dibujaRectangulos(Window ventana, GC el_GC, XRectangle *lasRectas, int elNumeroDeRectas);
void llenaRectangulos(Window ventana, GC el_GC, XRectangle *lasRectas, int elNumeroDeRectas);
void dibujaSegmentos(Window ventana, GC el_GC, XSegment segmentos[], int nsegmentos);
void llenaPoligono(Window ventana, GC el_GC, XPoint *puntos, int npuntos, int textura, int modo);
/*---------------------------------------------------------------*/
/* textx.c */
XFontStruct  *initFont(GC el_GC, char fontName[]);
void drawImageString(Window theWindow, GC el_GC, int x, int y, char string[]);
void drawString(Window theWindow, GC el_GC, int x, int y, char string[]);
/*---------------------------------------------------------------*/
/* windowx.c */
Window abroVentana(int x, int y, int width, int height, int flag, char titulo[],
                   int estado_icono, Window el_padre, GC *nuevo_GC);
int crea_GC(Window nueva_ventana, GC *nuevo_GC);
/*---------------------------------------------------------------*/
/* teclado.c */
int teclado_menu_inicial(XKeyEvent *theEvent);
int teclado_menu_modo(XKeyEvent *theEvent);
int teclado_menu_parametros(XKeyEvent *theEvent);
int teclado_menu_mostrar(XKeyEvent *theEvent);
int teclado_menu_grabacion(XKeyEvent *theEvent);
int teclado_botonOK(XKeyEvent *theEvent);
/*---------------------------------------------------------------*/
/* raton.c */
int raton_ayuda_modo(XButtonEvent *theEvent);
int raton_ayuda_grabacion(XButtonEvent *theEvent);
int raton_ayuda_mostrar(XButtonEvent *theEvent);
int raton_ayuda(XButtonEvent *theEvent);
/*---------------------------------------------------------------*/
/* cursores.c */
void cursores(void);
void limpiar_cursores(void);
/*---------------------------------------------------------------*/
/* dibujos.c */
void boton1(Window v, char letra[], char texto[], char color[], char color_oscuro[], char color_claro[]);
void boton2(Window v, char letra[], char texto[], char color[]);
void boton3(Window v, char color[], char color_fondo[]);
void boton4(Window v, char color[], char color_fondo[]);
void decorado_menu_inicial(Window v);
```

```
void expose_menu_inicial(void);
void expose_menu_modo(void);
void expose_menu_grabacion(void);
void expose_menu_parametros(int o1);
void expose_menu_mostrar(void);
void expose_menu_grabacion(void);
void expose_control(void);
void dibujo_entorno(Window v, char titulo[], char color_fondo[], int num_var,
                    char n1[], char n2[], char n3[], char n4[],
                    char c1[], char c2[], char c3[], char c4[], double val, double tmax);
void dibujo_punto(Window v, double x, double y, char color[]);
void pintar_caldera(Window v, int x, int y);
void pintar_quemador(Window v, int x, int y);
void pintar_valvula(Window v, int x, int y);
void pintar_supercalentador(Window v, int x, int y);
void pintar_turbina(Window v, int x, int y);
void pintar_condensador(Window v, int x, int y);
void pintar_bomba(Window v, int x, int y);
void pintar_fuente(Window v, int x, int y);
void pintar_flecha_arr(Window v, int x, int y, int l);
void pintar_flecha_abj(Window v, int x, int y, int l);
void pintar_flecha_izq(Window v, int x, int y, int l);
void pintar_flecha_der(Window v, int x, int y, int l);
/*-------------------------------------------------------------------*/
/* eventos.c */
void inicia_eventos(Window theWindow);
/*-------------------------------------------------------------------*/
/* ayudas.c */
void ayuda(void);
void expose_ayuda1(void);
void expose_ayuda2(void);
void expose_ayuda3(void);
void expose_ayuda(void);
void ayuda_modo(void);
void ayuda_grabacion(void);
void ayuda_mostrar(void);
/*-------------------------------------------------------------------*/
/* esquema1.c */
void abrir_controles_esquema1(void);
void cierro_controles(void);
void imprimo_controles_esquema1(double Qec, double Wlec, double Wvsc, double hlec);
void abrir_ventanas_esquema1(double tf);
void cierro_ventanas_esquema1(void);
/*-------------------------------------------------------------------*/
/* esquema2.c */
void abrir_controles_esquema2(void);
void imprimo_controles_esquema2(double Qec, double Qes, double Wlec, double Wvsc);
void abrir_ventanas_esquema2(double tf);
void cierro_ventanas_esquema2(void);
/*-------------------------------------------------------------------*/
/* esquema3.c */
void abrir_controles_esquema3(void);
void imprimo_controles_esquema3(double Qec, double Qes, double Tre, double Trs,
                                double Wvsc, double Wb, double Wr, double kx);
void abrir_ventanas_esquema3(double tf);
void cierro_ventanas_esquema3(void);
/*-------------------------------------------------------------------*/
/* clase Scheduler */

class Scheduler
{
private:
protected:
    char  nombre_display[120];
    int o1, o2, o3, o4, o5, o6;

    void abro_menu_inicial(void);
    int menu_inicial(void);
    int raton_menu_inicial(XButtonEvent *theEvent);

    void abro_menu_modo(void);
    int menu_modo(void);
    int raton_menu_modo(XButtonEvent *theEvent);
    void cierro_menu_modo(void);

    void abro_menu_parametros(void);
    int menu_parametros(void);
    int raton_menu_parametros(XButtonEvent *theEvent);
    void imprimo_parametros(void);
    void cierro_menu_parametros(void);

    void abro_menu_mostrar(void);
    int menu_mostrar(void);
    int raton_menu_mostrar(XButtonEvent *theEvent);
    void cierro_menu_mostrar(void);
    void set_fichero_mostrar(void);

    void muestra_simulacion(void);
```

```
    void esquema_mostrar(void);

    void abro_menu_grabacion(void);
    int menu_grabacion(void);
    int raton_menu_grabacion(XButtonEvent *theEvent);
    void set_grabacion(void);
    void creo_fichero(void);
    int raton_set_grabacion(XButtonEvent *theEvent);
    void cierro_menu_grabacion(void);

    void abro_control(void);
    void control(void);
    int  raton_control(XButtonEvent *theEvent);
    int  teclado_control(XKeyEvent *theEvent);
    int  raton_seguir(XButtonEvent *theEvent);
    void cierro_control(void);

    void esquema_1(void);
    void esquema_2(void);
    void esquema_3(void);

    double h, t, t0, tf, save;
    double P_caldera, P_condensador, W_refrigerante, T_refrig_in;
    double T_refrig_out, m_refrigerante, W_bomba, W_valvula1, kx_turbina;
    double h_supercal, m_supercal, k_supercal;

    double num_iter, hit, N_grupos_dat;
    int num_fich, file_mostrar;
    char filename[15];
    int fh;
    FILE *fp;

    double x, y;
    double x1=60., x2=250., y1=20., y2=170;

    void cambio_coordenadas(double u, double v, double u1, double v1)
        { x=x1+u*(x2-x1)/u1;  y=y2-v*(y2-y1)/v1; }

public:
    Scheduler () { }
    void start(void);
};
```


argsx.c

```
/* argsx.c */
/* CODIGO PARA MANEJAR ARGUMENTOS DE LINEA DE COMANDOS EN UN PROG. X */
/*=============================================*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/*=============================================*/
#define DISPLAY_DEFECTO  NULL                /* Valores por defecto */
#define GEOMETRIA_DEFECTO NULL
#define FONT_DEFECTO    "variable"
#define TITULO_DEFECTO   "CICLO DE VAPOR"
/*=============================================*/
void print_mensaje_ayuda(void)
    {
    fprintf(stderr,"The allowable command line options are:\n");
    fprintf(stderr,"\t-display displayname\n");
    fprintf(stderr,"\tUse a different display for output\n");
    fprintf(stderr,"\t-geometry geometryspec\n");
    fprintf(stderr,"\tSpecify window location and size\n");
    fprintf(stderr,"\t-font fontname\n");
    fprintf(stderr,"\tUse the given font name for text\n");
    fprintf(stderr,"\t-title windowtitle\n");
    fprintf(stderr,"\tUse the given name for the window title\n");
    fprintf(stderr,"\t-iconic\n");
    fprintf(stderr,"\tStart with the window in iconic state\n");
    }
/*=============================================*/
/* get_argumentos establece un juego de cadenas de texto, con valores por  */
/* defecto o con los entrados por el usuario mediante la linea de comandos*/
/****** argumentos:
    -h
    -help        imprime mensaje de ayuda
    -display usa el nombre del display que sigue al display sobre el que va el programa
    -geom
    -geometry    acepta localizaci<162n y/o tama<164o para la ventana
    -fn
    -font        usa el nombre del font que sigue, en lugar del por defecto
    -iconic  empieza un estado ic<162nico. Esta funci<162n devuelve 1 para un
             estado ic<162nico, 0 para estado normal
    -name
    -title       usa el texto que sigue como el ttulo de la ventana, mejor
```

```
            que el por defecto
******/
int get_argumentos(int argc,char *argv[], char  nombre_display[], char geometria[],
                   char  nombre_font[], char  titulo[])
    {
    int cont_argumentos;
    int estado_icono;

    nombre_display[0] = '\0';       /* selecciona valores por defecto */
    geometria[0]   = '\0';
    strcpy(nombre_font,FONT_DEFECTO);
    strcpy(titulo,TITULO_DEFECTO);
    estado_icono = 0;               /* estado no ic<162nico para empezar */
    for(cont_argumentos=0; cont_argumentos<argc; cont_argumentos++)
        {
        if(strncmp(argv[cont_argumentos],"-h",2)==0)/* comprueba si el usuario*/
            {                                      /* quiere mensaje de ayuda*/
            print_mensaje_ayuda(); exit(1);
            }
        if(strncmp(argv[cont_argumentos],"-display",8)==0)  /* comprueba si se */
            {                                               /* quiere un display*/
            cont_argumentos++;                              /* diferente */
            if(cont_argumentos<argc)
                { strcpy(nombre_display, argv[cont_argumentos]); }
            else
                { fprintf(stderr,"ERROR:the -display option should be %s\n",
                                "followed by a display name."); }
            }
        if(strncmp(argv[cont_argumentos],"-geom",5)==0)/* comprueba si quiere*/
            {                                         /* especificar tama<164o */
            cont_argumentos++;                        /* y/o localizaci<162n  */
            if(cont_argumentos<argc)                  /* para la ventana */
                { strcpy(geometria, argv[cont_argumentos]); }
            else
                {
                fprintf(stderr,"ERROR:the -geometry option should be %s\n",
                        "followed by a geometry spec,");
                fprintf(stderr,"e.g, 100x100+200+200 for \n%s\n%s\n",
                        "location 100,100","size 200 by 200.");
                }
            }
                    /* comprueba si se quiere ttulo para la ventana */
        if((strncmp(argv[cont_argumentos],"-title",6)==0)||(strncmp(argv[cont_argumentos],"name",5)==0))
            {
            cont_argumentos++;
            if(cont_argumentos<argc)
                { strcpy(titulo,argv[cont_argumentos]); }
            else
                { fprintf(stderr,"ERROR:the -title option should be %s\n", "followed by a window title."); }
            }
                    /* comprueba si se quiere empezar el prog. como un icono */
        if(strncmp(argv[cont_argumentos],"-iconic",7) == 0)
            { estado_icono = 1; }
    }
    if(strlen(nombre_display)<1)      nombre_display = NULL;
    if(strlen(geometria)<1)             geometria = NULL;
    return(estado_icono);
    }
```

ayudas.c

```
/** ayudas.c **/
/*=============================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>

#include "windows.h"
/*=============================================*/
extern Display   *el_display;
extern GC        el_GC;
extern Window    ventana_raiz;

extern Window       wa, wa0, wa1, wa2, wa3, waa, wbaa;
extern Window       wam, wbam;
extern Window       wmm;
extern Window       wget, wgeb, wg;
extern Window       wmtat, wmtab, wmt;

extern XFontStruct      *fontStruct, *initFont();
/*=============================================*/
void ayuda(void)
```

```
{
char        Titulo[120];
XEvent Event;
int       op, op2;

wa=abroVentana(100, 340, 600, 500, VENTANA_POP_UP, "Ayuda", ESTADO_NORMAL,
                    ventana_raiz,&el_GC);
inicia_eventos(wa);
wa0=abroVentana(550, 440, 40, 40, VENTANA_POP_UP, "wa0", ESTADO_NORMAL, wa, &el_GC);
inicia_eventos(wa0);
wa1=abroVentana(10, 450, 90, 40, VENTANA_POP_UP, "wa1", ESTADO_NORMAL, wa, &el_GC);
inicia_eventos(wa1);
wa2=abroVentana(110, 450, 90, 40, VENTANA_POP_UP, "wa2", ESTADO_NORMAL, wa, &el_GC);
inicia_eventos(wa2);
wa3=abroVentana(210, 450, 90, 40, VENTANA_POP_UP, "wa3", ESTADO_NORMAL, wa, &el_GC);
inicia_eventos(wa3);

expose_ayuda();

do  {
    XNextEvent(el_display, &Event);
    switch(Event.type)
        {
        //case KeyPress:
        case ButtonPress:    op=raton_ayuda(&Event.xbutton);break;
        default:             op=10;                         break;
        }
    XFlush(el_display);
    if(op==1 || op==2 || op==3)
        {
        waa=abroVentana(350, 150, 610, 600, VENTANA_POP_UP, "waa", ESTADO_NORMAL,
                    ventana_raiz,&el_GC);
        inicia_eventos(wa);
        wbaa=abroVentana(550, 550, 40, 40, VENTANA_POP_UP, "wbaa", ESTADO_NORMAL,
                    waa, &el_GC);
        inicia_eventos(wbaa);

        boton2(wbaa, "O", "k", "Orange");
        XFlush(el_display);

        if(op==1)expose_ayuda1();
        if(op==2)expose_ayuda2();
        if(op==3)expose_ayuda3();
        do  {
            XNextEvent(el_display,&Event);
            switch(Event.type)
                {
                //case KeyPress:
                case ButtonPress:    if(Event.xany.window == wbaa)op2=0;
                                     else                         op2=10;
                                     break;
                default: op2=10;break;
                }
            XFlush(el_display);
            }
            while(op2!=0);
        XDestroySubwindows(el_display,waa);
        XDestroyWindow(el_display,waa);
        }
    }
    while(op!=0);

XDestroySubwindows(el_display, wa);
XDestroyWindow(el_display, wa);
XFlush(el_display);
}
/*=========================================*/
void expose_ayuda(void)
{
set_color_nombre(el_GC, "Blue");
llenaRectangulo(wa,el_GC, 5, 5, 590, 490);

boton1(wa1, "1", "Planta1", "LimeGreen", "DarkGreen", "MediumSpringGreen");
boton1(wa2, "2", "Planta2", "LimeGreen", "DarkGreen", "MediumSpringGreen");
boton1(wa3, "3", "Planta3", "LimeGreen", "DarkGreen", "MediumSpringGreen");
boton2(wa0, "O", "k", "Orange");

set_color_nombre(el_GC, "Yellow");
fontStruct=initFont(el_GC, "variable");
drawString(wa, el_GC, 250, 20, "INTRODUCCION");
drawString(wa,el_GC,40,40,"Este programa simula el comportamiento del agua en unos sistemas
            (plantas) concretos. Se");
drawString(wa,el_GC,20,55,"han creado tres sistemas; los dos primeros son sucesivas construcciones
             no cerradas de lo que");
drawString(wa,el_GC,20,70,"sera la planta ciclica (tercer sistema). Por comodidad, llamamos
             plantas a las tres cuando real-");
drawString(wa,el_GC,20,85,"mente solo es la tercera. La ventaja de simular PLANTA1 y PLANTA2
            es que podemos ir viendo");
```

```
drawString(wa,el_GC,20,100,"c<162mo el ciclo evoluciona a medida que le sumamos dispositivos y lo
        vamos cerrando, hasta que");
drawString(wa,el_GC,20,115,"llegamos a PLANTA3. La motivaci<162n de todo <130sto es, pues,
        did<160ctica.");
drawString(wa,el_GC,40,130,"El programa permite guardar/recuperar simulaciones en/desde
        ficheros de datos.");
drawString(wa,el_GC,40,145,"En la simulaci<162n podemos distinguir cinco clases de variables:");
drawString(wa,el_GC,20,160,">> Variables que reflejan propiedades, magnitudes, etc.
        Todas son de este tipo.");
drawString(wa,el_GC,20,175,">> Unas cuantas de ellas necesitan tener inicializados sus valores en t0.");
drawString(wa,el_GC,20,190,">> De estas <163ltimas, algunas podemos definir sus valores iniciales
        en el 'men<163 de parametros':");
drawString(wa,el_GC,40,205,"-- t0 = Instante inicial (s)");
drawString(wa,el_GC,40,220,"-- tf = Instante final  (s)");
drawString(wa,el_GC,40,235,"-- h  = periodo de c<160lculo (s)");
drawString(wa,el_GC,40,250,"-- Adem<160s, cada planta cuenta con unos par<160metros adicionales.");
drawString(wa,el_GC,20,265,">> Un cuarto grupo  son las que se monitorizan en gr<160ficas.");
drawString(wa,el_GC,20,280,">> Por <163ltimo, unas variables de control permitir<160n
        interaccionar en  tiempo de simulaci<162n, con");
drawString(wa,el_GC,20,295," lo cual podremos observar los comportamientos ante determinadas
        variaciones.");
drawString(wa,el_GC,40,310,"Cada men<163 tiene su propia ventana de ayuda.");
drawString(wa,el_GC,20,325,"");
drawString(wa,el_GC,20,340,"");
drawString(wa,el_GC,20,355,"");
XFreeFont(el_display, fontStruct);
XFlush(el_display);
}
/*=========================================*/
void expose_ayuda1(void)
{
pintar_caldera(waa,250,170);
pintar_flecha_der(waa,250,245,30);      pintar_quemador(waa,155,235);
pintar_flecha_arr(waa,280,110,60);      pintar_valvula(waa,280,100);
pintar_flecha_arr(waa,280,320,60);      pintar_valvula(waa,280,390);
pintar_flecha_izq(waa,290,390,60);      pintar_fuente(waa,350,380);
XFlush(el_display);
}
/*=========================================*/
void expose_ayuda2(void)
{
pintar_caldera(waa,200,170);
pintar_flecha_der(waa,200,245,30);      pintar_quemador(waa,105,235);
pintar_flecha_arr(waa,230,110,60);      pintar_valvula(waa,230,100);
pintar_flecha_der(waa,300,100,60);      pintar_supercalentador(waa,300,80);
pintar_flecha_arr(waa,360,120,40);      pintar_quemador(waa,330,160);
pintar_flecha_der(waa,500,100,80);      pintar_valvula(waa,500,100);
pintar_flecha_arr(waa,230,320,60);      pintar_valvula(waa,230,390);
pintar_flecha_izq(waa,240,390,60);      pintar_fuente(waa,300,380);
XFlush(el_display);
}
/*=========================================*/
void expose_ayuda3(void)
{
pintar_caldera(waa,100,170);
pintar_flecha_der(waa,100,245,30);      pintar_quemador(waa,5,235);
pintar_flecha_arr(waa,130,110,60);      pintar_valvula(waa,130,100);
pintar_flecha_arr(waa,130,320,70);
pintar_flecha_der(waa,200,100,60);      pintar_supercalentador(waa,200,80);
pintar_flecha_arr(waa,260,120,40);      pintar_quemador(waa,230,160);
pintar_flecha_der(waa,460,100,140);     pintar_turbina(waa,460,100);
pintar_flecha_abj(waa,480,210,90);      pintar_condensador(waa,440,210);
pintar_flecha_abj(waa,480,360,40);      pintar_flecha_izq(waa,320,360,160);
pintar_bomba(waa,290,375);              pintar_flecha_izq(waa,130,390,130);
XFlush(el_display);
}
/*=========================================*/
void ayuda_modo(void)
{
XEvent Event;
int     op;

wam=abroVentana(10,150,230,140,VENTANA_NORMAL,"Ayuda",ESTADO_NORM,wmm,&el_GC);
inicia_eventos(wam);
wbam=abroVentana(190,240,40,40,VENTANA_NORMAL,"Ayuda",ESTADO_NORM,wmm,&el_GC);
inicia_eventos(wbam);

boton2(wbam,"O","K","Red");

fontStruct=initFont(el_GC,"variable");
drawString(wam,el_GC,10,20,"SIMULAR : Efectuamos una simula-");
drawString(wam,el_GC,80,30,"ci<162n de la planta.");
drawString(wam,el_GC,10,50,"MOSTRAR: Mostramos  una simula-");
drawString(wam,el_GC,80,60,"ci<162n  guardada  en  un");
drawString(wam,el_GC,80,70,"fichero.");


do  {
```

```
    XNextEvent(el_display,&Event);
    switch(Event.type)
        {
        case ButtonPress:   op=raton_ayuda_modo(&Event.xbutton);break;
        case KeyPress:      op=teclado_botonOK(&Event.xkey); break;
        default:            op=1;                            break;
        }
    }
    while(op!=0);

XFreeFont(el_display,fontStruct);
XFlush(el_display);

XDestroyWindow(el_display,wbam);
XDestroyWindow(el_display,wam);
XFlush(el_display);
}
/*=========================================*/
void ayuda_grabacion(void)
{
XEvent Event;
int    op;

wget=abroVentana(110,50,280,240,VENTANA_NORMAL,"wget",ESTADO_NORMAL,wg,&el_GC);
inicia_eventos(wget);
wgeb=abroVentana(340,240,40,40,VENTANA_NORMAL,"wgeb",ESTADO_NORMAL,wg,&el_GC);
inicia_eventos(wgeb);

boton2(wgeb,"O","K","Red");

fontStruct=initFont(el_GC,"variable");
drawString(wget,el_GC,10,16,"Si deseamos grabar, entonces los datos de");
drawString(wget,el_GC,10,30,"la simulaci<162n se grabar<160n  en el fichero");
drawString(wget,el_GC,10,44,"'simx_y.dat', donde 'x' es el n<163mero del");
drawString(wget,el_GC,10,58,"esquema, e 'y' es el n<163mero del fichero o de");
drawString(wget,el_GC,10,72,"lasimulaci<162n.");
drawString(wget,el_GC,10,86,"En este caso, se tomar<160n datos para grabar");
drawString(wget,el_GC,10,100,"cada 'hit' segundos, con  lo  cual tendremos");
drawString(wget,el_GC,10,120,"              'num_iter'");
drawString(wget,el_GC,10,134,"'N_dat' = ---------    grupos de datos a");
drawString(wget,el_GC,10,148,"              'hit'");
drawString(wget,el_GC,10,165,"grabar, donde  'num_iter '  es el numero de");
drawString(wget,el_GC,10,179,"iteraciones de la simulaci<162n:");
drawString(wget,el_GC,10,197,"              tf - t0");
drawString(wget,el_GC,10,211,"'num_iter' = ------");
drawString(wget,el_GC,10,225,"              h");

do  {
    XNextEvent(el_display,&Event);
    switch(Event.type)
        {
        case KeyPress:      op=teclado_botonOK(&Event.xkey);break;
        case ButtonPress:   op=raton_ayuda_grabacion(&Event.xbutton);break;
        default:            op=1;                                    break;
        }
    }
    while(op!=0);

XFreeFont(el_display,fontStruct);
XFlush(el_display);

XDestroyWindow(el_display,wgeb);
XDestroyWindow(el_display,wget);
XFlush(el_display);
}
/*=========================================*/
void ayuda_mostrar(void)
{
XEvent Event;
int    op;

wmtat=abroVentana(10,150,200,90,VENTANA_NORMAL,"wmtat",ESTADO_NORML,wmt,&el_GC);
inicia_eventos(wmtat);
wmtab=abroVentana(165,195,40,40,VENTANA_NORMAL,"wmtab",ESTADO_NORM,wmt,&el_GC);
inicia_eventos(wmtab);

boton2(wmtab,"O","K","Red");

fontStruct=initFont(el_GC,"variable");
drawString(wmtat,el_GC,10,16,"Seleccione el fichero simulaci<162n");
drawString(wmtat,el_GC,10,30,"disponible simx_y.dat, donde :");
drawString(wmtat,el_GC,10,58,"x = n<163mero de esquema");
drawString(wmtat,el_GC,10,72,"y = num. de simulacion");

do  {
    XNextEvent(el_display,&Event);
    switch(Event.type)
        {
```

```
        case ButtonPress:    op=raton_ayuda_mostrar(&Event.xbutton);break;
        case KeyPress:       op=teclado_botonOK(&Event.xkey);break;
        default:             op=1;                          break;
        }
    }
    while(op!=0);

XFreeFont(el_display,fontStruct);
XFlush(el_display);

XDestroyWindow(el_display,wmtab);
XDestroyWindow(el_display,wmtat);
XFlush(el_display);
}
```

clases.c

```
/* clases.c */
/*=========================================*/
#include <stdio.h>
#include <math.h>

#include "clases.h"
/*=========================================*/
void Turbina::salida(void)
    {
    Ri=.5;
    Wtx=kx*Wvet;
    Wvst=Wvet-Wtx;

    vap_sal_i.setPys(Pvst, svet);       /* Cálculo de hvsti */
    hvsti=vap_sal_i.dar_h();

    hvst=hvsti-Ri*(hvet-hvsti);

    vap_sal.setPyh(Pvst,hvst)              /*Cálculo de Tvst,svst,xvst,vvst*/
    Tvst=vap_sal.dar_T();          xvst=vap_sal.dar_x();
    vvst=vap_sal.dar_v();          svst=vap_sal.dar_s();
    estado_salida=vap_sal.dar_estado();
    }
/*=========================================*/
void Condensador::set1_variables(void)
    {
    k2=0.;k4=0.;
    for(int i=1;i<=orden_interp_sat;i++)
        {
        k2=k2+i*b_sat[i]*pow(Pcd,i-1);
        k4=k4+i*d_sat[i]*pow(Pcd,i-1);
        }
    Vlcd=PI*ncd*pow(Rcd,2);            Vvcd=PI*pow(Rcd,2)*(Lcd-ncd);
    liq_cond.set_pyV(Pcd,Vlcd);
    dlcd=liq_cond.dar_densidad();      hlcd=liq_cond.dar_entalpia();
    hg=tabla_1d(Pcd,h_sat_v,P_sat,76);
    hl=tabla_1d(Pcd,h_sat_l,P_sat,76);
    vg=tabla_1d(Pcd,v_sat_v,P_sat,76);
    vl=tabla_1d(Pcd,v_sat_l,P_sat,76);
    }
/*-------------------------------------------------------------------*/
void Condensador::set2_variables(double Win, double T, double x, double h, double v, double Wout)
    {
    int i;
    k31=0.0; k11=0.0; k32=0.0; k12=0.0; k5=0.0;
    Wlscd=Wout;
    /*------------ vapor de entrada ---------------*/
    Wvecd=Win;  Tvecd=T;  xvecd=x;  hvecd=h;  dvecd=1./v;  Tcd=Tvecd;

    for(i=0;i<=orden_interp_vap;i++)     /* cálculo de c_x[] y a_x[] */
        {
        c_x[i]=tabla_1d(xvecd,c_vap[i],x_dat,11);
        a_x[i]=tabla_1d(xvecd,a_vap[i],x_dat,11);
        }
    for(i=1;i<=orden_interp_vap;i++)
        {
        k31=k31+i*c_x[i]*pow(Pcd,i-1);
        k11=k11+i*a_x[i]*pow(Pcd,i-1);
        }
    mvecd=Vvcd*dvecd;
    /*------------ vapor intermedio ---------------*/
    xm=xvecd/2.;          Tm=Tcd;
    hm=hl+xm*(hg-hl);  dm=1./(vl+xm*(vg-vl));
    for(i=0;i<=orden_interp_vap;i++)     /* cálculo de c_x[] y a_x[] */
        {
        c_x[i]=tabla_1d(xm,c_vap[i],x_dat,11);
```

```
        a_x[i]=tabla_1d(xm,a_vap[i],x_dat,11);
        }
    for(i=1;i<=orden_interp_vap;i++)
        {
        k12=k12+i*a_x[i]*pow(Pcd,i-1);
        k32=k32+i*c_x[i]*pow(Pcd,i-1);
        }
    mm=Vvcd*dm;
    /*---------- refrigerante ----------*/
    hre=tabla_2d_xy_liq(P_l, 27, arrT_l, 22, arrh_l, Pcd, Tre);
    dre=1./tabla_2d_xy_liq(P_l, 27, arrT_l, 22, arrv_l, Pcd, Tre);
    drs=1./tabla_2d_xy_liq(P_l, 27, arrT_l, 22, arrv_l, Pcd, Trs);
    hrs=tabla_2d_xy_liq(P_l, 27, arrT_l, 22, arrh_l, Pcd, Trs);

    for(i=0;i<=orden_interp_liq;i++)      /* c<160lculo de e_T[] */
        { e_T[i]=tabla_1d(Trs,e_liq[i],T_dat,22); }
    for(i=1;i<=orden_interp_liq;i++)
        { k5=k5+i*e_T[i]*pow(Pcd,i-1); }
    /*----------*/
    A0=PI*Rcd*Rcd;
    A1=dlcd-dvecd-dm;
    A2=Vlcd*k2+Vvcd*k11+Vvcd*k12;
    A3=hlcd*dlcd-hvecd*dvecd-dm*hm;
    A4=k11*hvecd*Vvcd+k12*hm*Vvcd+k2*hlcd*Vlcd+k31*dvecd*Vvcd+k32*dm*Vvcd+
        k4*dlcd*Vlcd;
    A4p=A4+mr*k5;
    A=((-hvecd+A4p/A2)*Wvecd+(hlcd-A4p/A2)*Wlscd-Wr*(hre-hrs))/(-A0*A3+A0*A1*A4p/A2);
    B=((-hvecd+A3/A1)*Wvecd+(hlcd-A3/A1)*Wlscd-Wr*(hre-hrs))/(-A4p+A2*A3/A1);
    }
/*----------*/
void Condensador::inicializo_ncdyPcd(double h)
    {
    ncd=ncd+A*h;
    Pcd=Pcd+B*h;
    }
/*==========*/
void Liquido_saturado::set_pyV(double p, double V)
    {
    presion=p;
    Volumen=V;
    temperatura=tabla_1d(presion, T_sat, P_sat, 76);
    volumen_especifico=tabla_1d(presion, v_sat_l, P_sat, 76);
    entalpia=tabla_1d(presion, h_sat_l, P_sat, 76);
    entropia=tabla_1d(presion, s_sat_l, P_sat, 76);
    densidad=1./volumen_especifico;
    masa=densidad*Volumen;
    }
/*==========*/
void Vapor_saturado::set_pyV(double p, double V)
    {
    presion=p;
    Volumen=V;
    temperatura=tabla_1d(presion, T_sat, P_sat, 76);
    volumen_especifico=tabla_1d(presion, v_sat_v, P_sat, 76);
    entalpia=tabla_1d(presion, h_sat_v, P_sat, 76);
    entropia=tabla_1d(presion, s_sat_v, P_sat, 76);
    densidad=1./volumen_especifico;
    masa=densidad*Volumen;
    }
/*==========*/
void Caldera::calculos_t(void)
    {
    k1=0.; k2=0.; k3=0.; k4=0.;
    for(int i=1;i<=orden_interp_sat;i++)
        {
        k1=k1+i*a_sat[i]*pow(Pc,i-1)
        k2=k2+i*b_sat[i]*pow(Pc,i-1);
        k3=k3+i*c_sat[i]*pow(Pc,i-1);
        k4=k4+i*d_sat[i]*pow(Pc,i-1);
        }
    vapcal.set_pyV(Pc, PI*Rc*Rc*(Lc-nc));
    liqcal.set_pyV(Pc, PI*Rc*Rc*nc);
    dvc=vapcal.dar_densidad();      dlc=liqcal.dar_densidad();
    hvc=vapcal.dar_entalpia();      hlc=liqcal.dar_entalpia();
    vvc=vapcal.dar_Volumen();       vlc=liqcal.dar_Volumen();
    svc=vapcal.dar_entropia();      slc=liqcal.dar_entropia();
    mvc=vapcal.dar_masa();          mlc=liqcal.dar_masa();
    mc=mvc+mlc;                     tc=liqcal.dar_temperatura();
    A0=PI*Rc*Rc;
    A1=dlc-dvc;
    A2=vlc*k2+vvc*k1;
    A3=hlc*dlc-hvc*dvc;
    A4=k1*hvc*vvc+k2*hlc*vlc+k3*dvc*vvc+k4*dlc*vlc;
    A=(Wvsc*(-hvc+A4/A2)-Wlec*(-hlec+A4/A2)+Qec)/(A3*A0-A0*A1*A4/A2);
    B=(Wvsc*(-hvc+A3/A1)-Wlec*(-hlec+A3/A1)+Qec)/(A4-A3*A2/A1);
    }
//----------//
void Caldera::inicializo_ncyPc(double h)
```

```
    {
    nc=nc+A*h;
    Pc=Pc+B*h;
    }
/*=============================================*/
void Vapor::setPyh(double P, double h)
    {
    double vg, vl, hg, hl, sg, sl;

    presion=P;
    entalpia=h;
    double hsat=tabla_1d(presion, h_sat_v, P_sat, 76);

    if(entalpia>hsat)      num_estado=2;    /* 2 = vapor supercalentado */
    else if(entalpia<hsat) num_estado=0;    /* 0 = vapor */
    else                   num_estado=1;    /* 1 = vapor saturado */
    switch(num_estado)
        {
        case 2: temperatura=tabla_2d_xz_sh(P_sup, 28, arrTK_sup, 18, arrh_sup, presion,
                                           entalpia)-273.15;
                volumen_especifico=tabla_2d_xy_sh(P_sup, 28, arrTK_sup, 18, arrv_sup,
                                           presion, temperatura+273.15);
                entropia=tabla_2d_xy_sh(P_sup, 28, arrTK_sup, 18, arrs_sup, presion,
                                           temperatura+273.15);
                densidad=1./volumen_especifico;
                estado="supercalentado";
                break;
        case 0: hg=tabla_1d(presion, h_sat_v, P_sat, 76);
                hl=tabla_1d(presion, h_sat_l, P_sat, 76);
                calidad=(entalpia-hl)/(hg-hl);
                temperatura=tabla_1d(presion, T_sat, P_sat, 76);
                    vg=tabla_1d(presion, v_sat_v, P_sat, 76);
                    vl=tabla_1d(presion, v_sat_l, P_sat, 76);
                volumen_especifico=vl+calidad*(vg-vl);
                    sg=tabla_1d(presion, s_sat_v, P_sat, 76);
                    sl=tabla_1d(presion, s_sat_l, P_sat, 76);
                entropia=sl+calidad*(sg-sl);
                densidad=1./volumen_especifico;
                estado="vapor";
                break;
        case 1: temperatura=tabla_1d(presion, T_sat, P_sat, 76);
                volumen_especifico=tabla_1d(entalpia, v_sat_v, h_sat_v, 76);
                entropia=tabla_1d(entalpia, s_sat_v, h_sat_v, 76);
                densidad=1./volumen_especifico;
                estado="saturado";
                break;
        default: break;
        }
    }
/*---------------------------------------------*/
void Vapor::setPys(double P, double s)
    {
    double vg, vl, hg, hl, sg, sl;

    presion=P;
    entropia=s;
    double ssat=tabla_1d(presion, s_sat_v, P_sat, 76);

    if(entropia>ssat)      num_estado=2;        /* 2 = vapor supercalentado */
    else if(entropia<ssat) num_estado=0;        /* 0 = vapor */
    else                   num_estado=1;        /* 1 = vapor saturado */
    switch(num_estado)
        {
        case 2: entalpia=tabla_2d_xy_sh(P_sup, 28, arrs_sup, 18, arrh_sup, presion, entropia);
                estado="supercalentado";
                break;
        case 0: sg=tabla_1d(presion, s_sat_v, P_sat, 76);
                sl=tabla_1d(presion, s_sat_l, P_sat, 76);
                calidad=(entropia-sl)/(sg-sl);
                hg=tabla_1d(presion, h_sat_v, P_sat, 76);
                hl=tabla_1d(presion, h_sat_l, P_sat, 76);
                entalpia=hl+calidad*(hg-hl);
                estado="vapor";
                break;
        case 1: entalpia=tabla_1d(presion, h_sat_v, P_sat, 76);
                estado="saturado";
                break;
        default: break;
        }
    }
/*=============================================*/
void Supercalentador::calculos_t(double t, double W)
    {
    Pvss=Pves-ksup*pow(Wves,2)*vves;
    Wvss=W;

    vapor_salida.setPyh(Pvss, hvss);
    Tvss=vapor_salida.dar_T();
```

```
    vvss=vapor_salida.dar_v();
    svss=vapor_salida.dar_s();
    estado_salida=vapor_salida.dar_estado();

    k=(hvss-(Qes+hves*Wves)/Wves)*exp(Wves*t/mvs);
    }
/*-------------------------------------------------------------------*/
void Supercalentador::inicializo_hvss(double t)
    {
    hvss=((Qes+hves*Wves)/Wves)+k*exp(-Wves*t/mvs);
    }
```

colorx.c

```
/* colorx.c */      /* INICIALIZACION DE LOS COLORES */

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <string.h>

#include "windows.h"
/*=========================================*/
extern Display          *el_display;
extern Colormap         mapa_color;
extern int              profundidad;
extern unsigned long    pixel_negro;
extern unsigned long    pixel_blanco;

#define pixels_max 66

unsigned long los_pixels[pixels_max];

char *nombres_colores[pixels_max]=
{
    "Aquamarine",       /*0*/
    "Black",            /*1*/
    "Blue",             /*2*/
    "BlueViolet",       /*3*/
    "Brown",            /*4*/
    "CadetBlue",        /*5*/
    "Coral",            /*6*/
    "CornflowerBlue",   /*7*/
    "Cyan",             /*8*/
    "DarkGreen",        /*9*/
    "DarkOliveGreen",   /*10*/
    "DarkOrchid",       /*11*/
    "DarkSlateBlue",    /*12*/
    "DarkSlateGrey",    /*13*/
    "DarkTurquoise",    /*14*/
    "DimGrey",          /*15*/
    "FireBrick",        /*16*/
    "ForestGreen",      /*17*/
    "Gold",             /*18*/
    "Goldenrod",        /*19*/
    "Grey",             /*20*/
    "Green",            /*21*/
    "GreenYellow",      /*22*/
    "IndianRed",        /*23*/
    "Khaki",            /*24*/
    "LightBlue",        /*25*/
    "LightGrey",        /*26*/
    "LightSteelBlue",   /*27*/
    "LimeGreen",        /*28*/
    "Magenta",          /*29*/
    "Maroon",           /*30*/
    "MediumAquamarine", /*31*/
    "MediumBlue",       /*32*/
    "MediumForestGreen" /*33*/
    "MediumGoldenrod",  /*34*/
    "MediumOrchid",     /*35*/
    "MediumSeaGreen",   /*36*/
    "MediumSlateBlue",  /*37*/
    "MediumSpringGreen", /*38*/
    "MediumTurquoise",  /*39*/
    "MediumVioletRed",  /*40*/
    "MidNIghtBlue",     /*41*/
    "Navy",             /*42*/
    "Orange",           /*43*/
    "OrangeRed",        /*44*/
    "Orchid",           /*45*/
    "PaleGreen",        /*46*/
    "Pink",             /*47*/
    "Plum",             /*48*/
```

```
        "Red",              /*49*/
        "Salmon",           /*50*/
        "SeaGreen",         /*51*/
        "Sienna",           /*52*/
        "SkyBlue",          /*53*/
        "SlateBlue",        /*54*/
        "SpringGreen",      /*55*/
        "SteelBlue",        /*56*/
        "Tan",              /*57*/
        "Thistle",          /*58*/
        "Turquoise",        /*59*/
        "Violet",           /*60*/
        "VioleRed",         /*61*/
        "Wheat",            /*62*/
        "White",              /*63*/
        "Yellow",           /*64*/
        "YEllowGreen"       /*65*/
    };
/*=========================================*/
void set_color_nombre(GC el_GC,  char theName[])
    {
    int i;
    i=0;
    while((strcmp(theName, nombres_colores[i])!=0) && (i<pixels_max))
        { i++; }
    if(i<pixels_max)
        { XSetForeground(el_display, el_GC, los_pixels[i]); }
    }
/*=========================================*/
void init_colores_defecto(void)
    {
    XColor theRGBColor, theHardwareColor;
    int    theStatus;
    int    i;

    if(profundidad  1)
        {
        for(i=0;i<pixels_max;i++)
            {
            theStatus = XLookupColor(el_display, mapa_color,
                                nombres_colores[i], &theRGBColor, &theHardwareColor);
            if(theStatus != 0)
                {
                theStatus = XAllocColor( el_display, mapa_color, &theHardwareColor);
                if(theStatus != 0)
                    { los_pixels[i] = theHardwareColor.pixel; }
                }
            }
        }
    else
        {
        for(i=0;i<pixels_max;i++)
            {
            if(strcmp("white",nombres_colores[i])==0)
                { los_pixels[i] = pixel_blanco; }
            else
                { los_pixels[i] = pixel_negro; }
            }
        }
    }
/*=========================================*/
void set_color(GC el_GC,  int colorNumber)
    {
    if((colorNumber < pixels_max ) && (colorNumber >=0 ))
        { XSetForeground(el_display, el_GC, los_pixels[colorNumber]); }
    }
```

cursores.c

```
/* cursores.c */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/cursorfont.h>

#include "windows.h"
/*=========================================*/
Cursor      wCursor;
Cursor      w0Cursor;
Cursor      w1Cursor;
Cursor      w2Cursor;
Cursor      w3Cursor;
Cursor      w4Cursor;
extern Display   *el_display;
```

```
extern Window  w;
extern Window  w0; /* subwindows de theWindow*/
extern Window  w1;
extern Window  w2;
extern Window  w3;
extern Window  w4;
/*=========================================*/
void cursores(void)
    {
    wCursor = XCreateFontCursor(el_display, XC_X_cursor);
    w0Cursor = XCreateFontCursor(el_display, XC_pirate);
    w1Cursor = XCreateFontCursor(el_display, XC_hand2);
    w2Cursor = XCreateFontCursor(el_display, XC_hand2);
    w3Cursor = XCreateFontCursor(el_display, XC_hand2);
    w4Cursor = XCreateFontCursor(el_display, XC_gumby);

    XDefineCursor(el_display, w, wCursor);
    XDefineCursor(el_display, w0, w0Cursor);
    XDefineCursor(el_display, w1, w1Cursor);
    XDefineCursor(el_display, w2, w2Cursor);
    XDefineCursor(el_display, w3, w3Cursor);
    XDefineCursor(el_display, w4, w4Cursor);
    }
/*=========================================*/
void limpiar_cursores(void)
    {
    XFreeCursor(el_display, wCursor);
    XFreeCursor(el_display, w0Cursor);
    XFreeCursor(el_display, w1Cursor);
    XFreeCursor(el_display, w2Cursor);
    XFreeCursor(el_display, w3Cursor);
    XFreeCursor(el_display, w4Cursor);
    }
```


datos.c

```
/* datos.c */          /* DATOS PARA DIBUJOS */
/*=========================================*/
#include <X11/Xlib.h>
/*=========================================*/
/* boton1 */
XPoint puntos_marco_arriba[4]={ {0, 0}, {90, 0}, {85, 5}, {5, 5} };
XPoint puntos_marco_abajo[4]={ {5, 35}, {85, 35}, {90, 40}, {0, 40} };
XPoint puntos_marco_izq[4]={ {0, 0}, {5, 5}, {5, 35}, {0, 40} };
XPoint puntos_marco_dch[4]={ {90, 0}, {90, 40}, {85, 35}, {85, 0} };
/*=========================================*/
/* boton3 */
XPoint flecha_izquierda[3]={ {0, 10}, {20, 0}, {20, 20} };
/*=========================================*/
/* boton4 */
XPoint flecha_derecha[3]={ {0, 0}, {20, 10}, {0, 20} };
/*=========================================*/
/* decorado_menu_inicial */
XPoint pdmi0[8]={ {105, 370}, {380, 370}, {340, 380}, {490, 380},
          {450, 390}, {545, 390}, {545, 395}, {105, 395} };
XPoint pdmi1[11]={{105, 350}, {130, 350}, {190, 330}, {510, 330}, {545, 350},
          {545, 390}, {450, 390}, {490, 380}, {340, 380}, {380, 370}, {105, 370} };
XPoint pdmi2[4]={ {230, 330}, {230, 310}, {240, 300}, {240, 330} };
XPoint pdmi3[4]={ {330, 300}, {330, 260}, {340, 250}, {340, 300} };
XPoint pdmi4[4]={ {340, 250}, {340, 240}, {350, 230}, {350, 250} };
XPoint pdmi5[4]={ {420, 330}, {420, 310}, {430, 300}, {430, 330} };
XPoint pdmi6[4]={ {240, 330}, {240, 300}, {420, 300}, {420, 330} };
XPoint pdmi7[4]={ {340, 300}, {340, 250}, {420, 250}, {420, 300} };
XPoint pdmi8[4]={ {350, 250}, {350, 230}, {410, 230}, {410, 250} };
XPoint pdmi9[4]={ {430, 330}, {430, 300}, {510, 300}, {510, 330} };
XPoint pdmi10[6]={ {160, 340}, {170, 160}, {170, 110}, {180, 110}, {180, 160}, {190, 330} };
XPoint pdmi11[6]={ {190, 330}, {200, 150}, {200, 100}, {210, 100}, {210, 150}, {220, 330} };
XPoint pdmi12[6]={ {170, 335}, {173, 160}, {173, 110}, {170, 110}, {170, 160}, {160, 340} };
XPoint pdmi13[6]={ {200, 330}, {203, 150}, {203, 100}, {200, 100}, {200, 150}, {190, 330} };
XPoint pdmi14[4]={ {170, 160}, {170, 150}, {180, 150}, {180, 160} };
XPoint pdmi15[4]={ {170, 140}, {170, 130}, {180, 130}, {180, 140} };
XPoint pdmi16[4]={ {200, 150}, {200, 140}, {210, 140}, {210, 150} };
XPoint pdmi17[4]={ {200, 130}, {200, 120}, {210, 120}, {210, 130} };
```

dibujos.c

```
/* dibujos.c */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>

#include "windows.h"
/*=========================================*/
extern GC       el_GC;
extern Display  *el_display;

extern Window  w, w0, w1, w2, w3, w4;
extern Window  wmm, ws, wm, wc, we;
extern Window  wp, wpc, wpk, wp00, wp01, wp10, wp11, wp20, wp21, wp30, wp31, wp40, wp41;
extern Window      wp50, wp51, wp60, wp61, wp70, wp71;
extern Window      wp02, wp03, wp12, wp13, wp22, wp23, wp32, wp33, wp42, wp43;
extern Window      wp52, wp53, wp62, wp63;
extern Window  wg, wgsg, wgng, wgc, wgc;
extern Window  wcn, wcn1, wcn2, wcn3, wcn4, wd;
extern Window  wmt, wmtk, wmtc, wmta, wmt00, wmt01;

XFontStruct    *fontStruct, *initFont();
char           buffer[50];
/*=========================================*/
void boton1(Window v, char letra[], char texto[], char color[], char color_oscuro[], char color_claro[])
{
extern XPoint puntos_marco_arriba[4];
extern XPoint puntos_marco_abajo[4];
extern XPoint puntos_marco_izq[4];
extern XPoint puntos_marco_dch[4];

set_color_nombre(el_GC,color);
llenaRectangulo(v, el_GC, 5, 5, 80, 30);

set_color_nombre(el_GC,color_oscuro);
llenaPoligono(v, el_GC, puntos_marco_abajo, 4, 0, 0);
llenaPoligono(v, el_GC, puntos_marco_dch, 4, 0, 0);

set_color_nombre(el_GC,color_claro);
llenaPoligono(v, el_GC, puntos_marco_arriba, 4, 0, 0);
llenaPoligono(v, el_GC, puntos_marco_izq, 4, 0, 0);

set_color_nombre(el_GC,"Black");
fontStruct=initFont(el_GC,"variable");
drawImageString(v, el_GC, 20, 25, letra);
drawString(v, el_GC, 30, 25, texto);

XFlush(el_display);
XFreeFont(el_display,fontStruct);
}
/*=========================================*/
void boton3(Window v, char color[], char color_fondo[])
{
extern XPoint flecha_izquierda[3];

set_color_nombre(el_GC,color_fondo);
llenaRectangulo(v, el_GC, 0, 0, 20, 20);

set_color_nombre(el_GC,color);
XFillPolygon(el_display, v, el_GC, flecha_izquierda, 3, 0, 0);

set_color_nombre(el_GC,"Black");
fontStruct=initFont(el_GC,"variable");

drawString(v, el_GC, 10, 13, "-");
XFlush(el_display);
XFreeFont(el_display,fontStruct);
}
/*=========================================*/
void boton4(Window v, char color[], char color_fondo[])
{
extern XPoint flecha_derecha[3];

set_color_nombre(el_GC,color_fondo);
llenaRectangulo(v, el_GC, 0, 0, 20, 20);

set_color_nombre(el_GC,color);
XFillPolygon(el_display, v, el_GC, flecha_derecha, 3, 0, 0);

set_color_nombre(el_GC,"Black");
fontStruct=initFont(el_GC,"variable");

drawString(v, el_GC, 4, 13, "+");
XFlush(el_display);
XFreeFont(el_display,fontStruct);
}
```

```
/*=========================================*/
void boton2(Window v, char letra[], char texto[], char color[])
{
set_color_nombre(el_GC,color);
llenaOvalo(v, el_GC, 0, 0, 40, 40);

set_color_nombre(el_GC,"Black");

fontStruct=initFont(el_GC,"variable");

drawImageString(v, el_GC, 10, 20, letra);
drawString(v, el_GC, 20, 25, texto);

XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void decorado_menu_inicial(Window v)
{
extern XPoint pdmi0[8];        extern XPoint pdmi1[4];
extern XPoint pdmi2[4];        extern XPoint pdmi3[4];
extern XPoint pdmi4[4];        extern XPoint pdmi5[4];
extern XPoint pdmi6[4];        extern XPoint pdmi7[4];
extern XPoint pdmi8[4];        extern XPoint pdmi9[4];
extern XPoint pdmi10[6];           extern XPoint pdmi11[6];
extern XPoint pdmi12[6];       extern XPoint pdmi13[6];
extern XPoint pdmi14[4];       extern XPoint pdmi15[4];
extern XPoint pdmi16[4];       extern XPoint pdmi17[4];

set_color_nombre(el_GC,"DarkOliveGreen");
llenaRectangulo(v, el_GC, 105, 5, 440, 45);
fontStruct=initFont(el_GC,"variable");
drawImageString(w, el_GC, 180, 30, "CICLO DE VAPOR EN UNA PLANTA INDUSTRIAL");

set_color_nombre(el_GC,"SkyBlue");       llenaRectangulo(v, el_GC, 105, 55, 440, 340);
set_color_nombre(el_GC,"Blue");      llenaPoligono(v, el_GC, pdmi0, 8, 0, 0);
set_color_nombre(el_GC,"SeaGreen");  llenaPoligono(v, el_GC, pdmi1, 11, 0, 0);

set_color_nombre(el_GC,"Black");     llenaPoligono(v, el_GC, pdmi2, 4, 0, 0);
set_color_nombre(el_GC,"Black");     llenaPoligono(v, el_GC, pdmi3, 4, 0, 0);
set_color_nombre(el_GC,"Black");     llenaPoligono(v, el_GC, pdmi4, 4, 0, 0);
set_color_nombre(el_GC,"Black");     llenaPoligono(v, el_GC, pdmi5, 4, 0, 0);

set_color_nombre(el_GC,"Goldenrod"); llenaPoligono(v, el_GC, pdmi6, 4, 0, 0);
set_color_nombre(el_GC,"Coral");     llenaPoligono(v, el_GC, pdmi7, 4, 0, 0);
set_color_nombre(el_GC,"Goldenrod"); llenaPoligono(v, el_GC, pdmi8, 4, 0, 0);
set_color_nombre(el_GC,"Goldenrod"); llenaPoligono(v, el_GC, pdmi9, 4, 0, 0);

set_color_nombre(el_GC,"White");     llenaArco(v, el_GC, 250, 275, 70, 50, 0, 180*64);
set_color_nombre(el_GC,"Black");     llenaArco(v, el_GC, 250, 275, 70, 50, 90*64, 90*64);
set_color_nombre(el_GC,"White");     llenaArco(v, el_GC, 260, 275, 50, 50, 0, 180*64);

set_color_nombre(el_GC,"Grey");      llenaPoligono(v, el_GC, pdmi10, 6, 0, 0);
set_color_nombre(el_GC,"Grey");      llenaPoligono(v, el_GC, pdmi11, 6, 0, 0);
set_color_nombre(el_GC,"Yellow");    llenaPoligono(v, el_GC, pdmi14, 4, 0, 0);
set_color_nombre(el_GC,"Yellow");    llenaPoligono(v, el_GC, pdmi15, 4, 0, 0);
set_color_nombre(el_GC,"Yellow");    llenaPoligono(v, el_GC, pdmi16, 4, 0, 0);
set_color_nombre(el_GC,"Yellow");    llenaPoligono(v, el_GC, pdmi17, 4, 0, 0);
set_color_nombre(el_GC,"DimGrey");   llenaPoligono(v, el_GC, pdmi12, 6, 0, 0);
set_color_nombre(el_GC,"DimGrey");   llenaPoligono(v, el_GC, pdmi13, 6, 0, 0);

set_color_nombre(el_GC,"Black");
dibujaLinea(v, el_GC, 250, 310, 410, 310);
dibujaLinea(v, el_GC, 440, 310, 500, 310);

for(int i=0;i<=3;i++)
    dibujaLinea(v, el_GC, 340, 260+10*i, 420, 260+10*i);

drawString(v, el_GC, 280, 100, "JUAN ANTONIO GOMEZ PULIDO");
drawString(v, el_GC, 290, 120, "JOSE MARIA GIRON SIERRA");
drawString(v, el_GC, 250, 150, "Departamento de Informática y Automática");
drawString(v, el_GC, 290, 165, "Facultad de Ciencias Físicas");
drawString(v, el_GC, 250, 180, "Universidad Complutense de Madrid - 1992");

XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void expose_menu_inicial(void)
{
decorado_menu_inicial(w);
boton1(w0, "S", "alir", "Magenta", "DarkSlateBlue", "LightBlue");
boton1(w1, "1", "-Ciclo1", "Grey", "DarkSlateGrey", "LightGrey");
boton1(w2, "2", "-Ciclo2", "Grey", "DarkSlateGrey", "LightGrey");
boton1(w3, "3", "-Ciclo3", "Grey", "DarkSlateGrey", "LightGrey");
boton1(w4, "A", "yuda", "LimeGreen", "DarkGreen", "MediumSpringGreen");
XFlush(el_display);
}
```

```
/*=========================================*/
void expose_menu_modo(void)
{
set_color_nombre(el_GC,"Khaki");
llenaRectangulo(wmm, el_GC, 10, 10, 230, 30);
set_color_nombre(el_GC,"Black");

boton1(wc, "C", "ancelar", "Magenta", "DarkSlateBlue", "LightBlue");
boton1(ws, "S", "imular", "Grey", "DarkSlateGrey", "LightGrey");
boton1(wm, "M", "ostrar", "Grey", "DarkSlateGrey", "LightGrey");
boton1(we, "E", "xplicar", "LimeGreen", "DarkGreen", "MediumSpringGreen");

fontStruct=initFont(el_GC,"variable");
drawString(wmm, el_GC, 65, 30, "MENU DE MODO");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void expose_menu_parametros(int o1)
{
set_color_nombre(el_GC,"Khaki");
llenaRectangulo(wp, el_GC, 10, 10, 430, 30);
set_color_nombre(el_GC,"Black");
switch(o1)
    {
    case 3: boton3(wp40, "LightBlue", "Yellow");      boton4(wp41, "LightBlue", "Yellow");
            boton3(wp50, "LightBlue", "Yellow");      boton4(wp51, "LightBlue", "Yellow");
            boton3(wp60, "LightBlue", "Yellow");      boton4(wp61, "LightBlue", "Yellow");
            boton3(wp70, "LightBlue", "Yellow");      boton4(wp71, "LightBlue", "Yellow");
            boton3(wp02, "LightBlue", "Yellow");      boton4(wp03, "LightBlue", "Yellow");
            boton3(wp12, "LightBlue", "Yellow");      boton4(wp13, "LightBlue", "Yellow");
            boton3(wp22, "LightBlue", "Yellow");      boton4(wp23, "LightBlue", "Yellow");
            boton3(wp32, "LightBlue", "Yellow");      boton4(wp33, "LightBlue", "Yellow");
            boton3(wp42, "LightBlue", "Yellow");      boton4(wp43, "LightBlue", "Yellow");
            boton3(wp52, "LightBlue", "Yellow");      boton4(wp53, "LightBlue", "Yellow");
            boton3(wp62, "LightBlue", "Yellow");      boton4(wp63, "LightBlue", "Yellow");
    case 2:
    case 1: boton3(wp00, "LightBlue", "Yellow");      boton4(wp01, "LightBlue", "Yellow");
            boton3(wp10, "LightBlue", "Yellow");      boton4(wp11, "LightBlue", "Yellow");
            boton3(wp20, "LightBlue", "Yellow");      boton4(wp21, "LightBlue", "Yellow");
            boton3(wp30, "LightBlue", "Yellow");      boton4(wp31, "LightBlue", "Yellow");
            break;
    default: break;
    }
```



```
boton1(wpc, "C", "ancelar", "Magenta", "DarkSlateBlue", "LightBlue");
boton1(wpk, "O", "k", "Grey", "DarkSlateGrey", "LightGrey");

fontStruct=initFont(el_GC,"variable");
drawString(wp, el_GC, 150, 30, "MENU DE PARAMETROS");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void expose_menu_mostrar(void)
{
set_color_nombre(el_GC,"Khaki");
llenaRectangulo(wmt, el_GC, 10, 10, 200, 30);
set_color_nombre(el_GC,"Black");

dibujaRectangulo(wmt, el_GC, 50, 250, 120, 40);
dibujaRectangulo(wmt, el_GC, 50, 300, 120, 20);

boton1(wmtc, "C", "ancelar", "Magenta", "DarkSlateBlue", "LightBlue");
boton1(wmtk, "O", "k", "Grey", "DarkSlateGrey", "LightGrey");
boton1(wmta, "A", "yuda", "LimeGreen", "DarkGreen", "MediumSpringGreen");
boton3(wmt00, "LightBlue", "Yellow"); boton4(wmt01, "LightBlue", "Yellow");

fontStruct=initFont(el_GC,"variable");
drawString(wmt, el_GC, 50, 30, "MENU DE MOSTRAR");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void Scheduler::imprimo_parametros(void)
{
char buffer[50];

XClearArea(el_display, wp, 70, 50, 370, 230, 0);
switch(o1)
    {
    case 3: sprintf(buffer, "P_condensador=%-3.0f", P_condensador);
            drawString(wp, el_GC, 70, 185, buffer);
            sprintf(buffer, "W_refrigerante=%-5.3f", W_refrigerante);
            drawString(wp, el_GC, 70, 215, buffer);
            sprintf(buffer, "T_refrig_in=%-4.1f", T_refrig_in);
            drawString(wp, el_GC, 70, 245, buffer);
            sprintf(buffer, "T_refrig_out=%-4.1f", T_refrig_out);
            drawString(wp, el_GC, 70, 275, buffer);
```

```
            sprintf(buffer, "m_refrigerante=%-3.1f", m_refrigerante);
            drawString(wp, el_GC, 290, 65, buffer);
            sprintf(buffer, "W_bomba=%-5.4f", W_bomba);
            drawString(wp, el_GC, 290, 95, buffer);
            sprintf(buffer, "W_valvula1=%-5.4f", W_valvula1);
            drawString(wp, el_GC, 290, 125, buffer);
            sprintf(buffer, "kx_turbina=%-4.2f", kx_turbina);
            drawString(wp, el_GC, 290, 155, buffer);
            sprintf(buffer, "h_supercal=%-7.2f", h_supercal);
            drawString(wp, el_GC, 290, 185, buffer);
            sprintf(buffer, "m_supercal=%-6.2f", m_supercal);
            drawString(wp, el_GC, 290, 215, buffer);
            sprintf(buffer, "k_supercal=%-6.2f", k_supercal);
            drawString(wp, el_GC, 290, 245, buffer);
    case 2:
    case 1: sprintf(buffer, "t0=%-3.2f", t0);
            drawString(wp, el_GC, 70, 65, buffer);
            sprintf(buffer, "tf=%-8.1f", tf);
            drawString(wp, el_GC, 70, 95, buffer);
            sprintf(buffer, "h=%-6.4f", h);
            drawString(wp, el_GC, 70, 125, buffer);
            sprintf(buffer, "P_caldera=%-6.0f", P_caldera);
            drawString(wp, el_GC, 70, 155, buffer);
            break;
    default: break;
    }
}
/*=========================================*/
void expose_menu_grabacion(void)
{
set_color_nombre(el_GC,"Khaki");
llenaRectangulo(wg, el_GC, 10, 10, 380, 30);
set_color_nombre(el_GC,"Black");

boton1(wgc, "C", "ancelar", "Magenta", "DarkSlateBlue", "LightBlue");
boton1(wgsg, "S", "i grabar", "Grey", "DarkSlateGrey", "LightGrey");
boton1(wgng, "N", "o grabar", "Grey", "DarkSlateGrey", "LightGrey");
boton1(wge, "E", "xplicar", "LimeGreen", "DarkGreen", "MediumSpringGreen");

fontStruct=initFont(el_GC,"variable");
drawString(wg, el_GC, 120, 30, "MENU DE GRABACION");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
```



```
/*=========================================*/
void expose_control(void)
{
set_color_nombre(el_GC,"Khaki");
llenaRectangulo(wcn, el_GC, 10, 10, 330, 30);
set_color_nombre(el_GC,"Black");

boton1(wcn1, "E", "mpezar", "Grey", "DarkSlateGrey", "LightGrey");
boton1(wcn2, "P", "arar", "LimeGreen", "DarkGreen", "MediumSpringGreen");
boton1(wcn3, "S", "eguir", "LimeGreen", "DarkGreen", "MediumSpringGreen");
boton1(wcn4, "C", "ancelar", "Magenta", "DarkSlateBlue", "LightBlue");

fontStruct=initFont(el_GC,"variable");
drawString(wcn, el_GC, 140, 30, "CONTROL");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void dibujo_entorno(Window v, char titulo[], char color_fondo[], int num_var,
            char n1[], char n2[], char n3[], char n4[], char c1[], char c2[], char c3[],
            char c4[], double val, double tmax)
{
int i;
char buffer[20];

set_color_nombre(el_GC, color_fondo);    // Establezco el color del fondo
llenaRectangulo(v, el_GC, 3, 3, 274, 244);

set_color_nombre(el_GC, "Black");      // Dibujo los ejes
dibujaLinea(v, el_GC, 60, 20, 60, 170);
dibujaLinea(v, el_GC, 60, 170, 250, 170);

for(i=0;i<=10;i=i+1)
    dibujaLinea(v, el_GC, 60+i*19, 167, 60+i*19, 173); // eje x
for(i=0;i<=20;i=i+1)
    dibujaLinea(v, el_GC, int(60+i*9.5), 168, int(60+i*9.5), 172);
for(i=0;i<=10;i=i+1)
    dibujaLinea(v, el_GC, 57, 20+i*15, 63, 20+i*15);   // eje y
for(i=0;i<=20;i=i+1)
    dibujaLinea(v, el_GC, 58, int(20+i*7.5), 62, int(20+i*7.5));

fontStruct=initFont(el_GC, "variable");     // Título del gráfico
drawString(v, el_GC, 100, 17, titulo);
drawString(v, el_GC, 255, 175, "t");
```

```
switch(num_var)          // Identificación del número de variables
    {
    case 4: set_color_nombre(el_GC,c4);        dibujaLinea(v, el_GC, 170, 230, 230, 230);
            set_color_nombre(el_GC,"Black");   drawString(v, el_GC, 235, 235, n4);
    case 3: set_color_nombre(el_GC,c3);        dibujaLinea(v, el_GC, 170, 200, 230, 200);
            set_color_nombre(el_GC,"Black");   drawString(v, el_GC, 235, 205, n3);
    case 2: set_color_nombre(el_GC,c2);        dibujaLinea(v, el_GC, 60, 230, 120, 230);
            set_color_nombre(el_GC,"Black");   drawString(v, el_GC, 125, 235, n2);
    case 1: set_color_nombre(el_GC,c1);        dibujaLinea(v, el_GC, 60, 200, 120, 200);
            set_color_nombre(el_GC,"Black");   drawString(v, el_GC, 125, 205, n1);
            break;
    default: break;
    }

fontStruct=initFont(el_GC, "5x8");     // Dimensiones de los ejes
drawString(v, el_GC, 60, 190, "0");
sprintf(buffer, "%-8.0f", tmax);       drawString(v, el_GC, 250, 190, buffer);
sprintf(buffer, "%-8.0f", tmax/2.);    drawString(v, el_GC, 155, 190, buffer);

drawString(v, el_GC, 10, 170, "0");
sprintf(buffer, "%-.3f", val);   drawString(v, el_GC, 10, 20, buffer);
sprintf(buffer, "%-.3f", val/2.); drawString(v, el_GC, 10, 95, buffer);

fontStruct=initFont(el_GC, "variable");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void dibujo_punto(Window v, double x, double y, char color[])
{
set_color_nombre(el_GC,color);
dibujaPunto(v, el_GC, int(x), int(y));
}
/*=========================================*/
void pintar_caldera(Window v, int x, int y)
{
XPoint ppc0[9]={ {x, y+100}, {x+10, y+90}, {x+20, y+100}, {x+30, y+90},
                 {x+40, y+100}, {x+50, y+90}, {x+60, y+100}, {x+60, y+150}, {x, y+150} };

set_color_nombre(el_GC,"Black");       pintar_flecha_arr(v, x-10, y+90, 60);
set_color_nombre(el_GC,"Orange");      llenaRectangulo(v, el_GC, x, y, 60, 150);
set_color_nombre(el_GC,"Blue");        llenaPoligono(v, el_GC, ppc0, 9, 0, 0);

fontStruct=initFont(el_GC,"variable");
set_color_nombre(el_GC,"Black");
drawImageString(v, el_GC, x+15, y+65, "Pc Tc");
drawImageString(v, el_GC, x+10, y+35, "Caldera");
drawImageString(v, el_GC, x-15, y+125, "nc");

drawString(v, el_GC, x+40, y-15, "Wvsc");        drawString(v, el_GC, x+40, y+165, "Wlec");
drawString(v, el_GC, x-25, y+60, "Qec");         drawString(v, el_GC, x+5, y+165, "hlec");

XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void pintar_quemador(Window v, int x, int y)
{
set_color_nombre(el_GC,"Yellow");      llenaRectangulo(v, el_GC, x, y, 65, 20);
set_color_nombre(el_GC,"Black");       dibujaRectangulo(v, el_GC, x, y, 65, 20);

fontStruct=initFont(el_GC,"variable");
drawString(v, el_GC, x+2, y+15, "Quemador");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void pintar_valvula(Window v, int x, int y)
{
set_color_nombre(el_GC,"Blue");        llenaOvalo(v, el_GC, x-10, y-10, 20, 20);

fontStruct=initFont(el_GC,"variable");
set_color_nombre(el_GC,"Black");       drawImageString(v, el_GC, x-20, y-20, "Valvula");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void pintar_supercalentador(Window v, int x, int y)
{
set_color_nombre(el_GC, "Gold"); llenaRectangulo(v, el_GC, x, y, 120, 40);

fontStruct=initFont(el_GC,"variable");
set_color_nombre(el_GC,"Black");
drawString(v, el_GC, x+15, y+25, "Supercalentador");
drawImageString(v, el_GC, x+70, y+55, "Qes");
drawImageString(v, el_GC, x-38, y-10, "Pves");   drawImageString(v, el_GC, x+130, y-10, "Pvss");
drawImageString(v, el_GC, x-38, y+5, "Tves");    drawImageString(v, el_GC, x+130, y+5, "Tvss");
drawImageString(v, el_GC, x-38, y+20, "vves");   drawImageString(v, el_GC, x+130, y+20, "vvss");
```

```
drawImageString(v, el_GC, x-38, y+35, "hves");   drawImageString(v, el_GC, x+130, y+35, "hvss");
drawImageString(v, el_GC, x-38, y+50, "sves");   drawImageString(v, el_GC, x+130, y+50, "svss");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void pintar_turbina(Window v, int x, int y)
{
XPoint ppt[4]={{x, y-10}, {x+60, y-40}, {x+60, y+40}, {x, y+10}};

set_color_nombre(el_GC, "DarkGreen");      llenaPoligono(v, el_GC, ppt, 4, 0, 0);
set_color_nombre(el_GC, "Black");          pintar_flecha_arr(v, 520, 20, 40);
llenaOvalo(v, el_GC, 515, 35, 10, 10);

fontStruct=initFont(el_GC,"variable");
drawImageString(v, el_GC, x+10, y+5, "Turbina");
drawString(v, el_GC, x+80, y-70, "Wtx");
drawString(v, el_GC, x+75, y-55, "(kx)");
drawImageString(v, el_GC, x-15, y+50, "Wvst  hvst");
drawImageString(v, el_GC, x-15, y+65, "vvst  Tvst");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void pintar_condensador(Window v, int x, int y)
{
XPoint ppc[20]={{x+100, y+10}, {x+10, y+10}, {x+10, y+40}, {x+60, y+40},
                {x+60, y+50}, {x+10, y+50}, {x+10, y+80}, {x+60, y+80},
                {x+60, y+90}, {x-20, y+90}, {x-20, y+100}, {x+70, y+100},
                {x+70, y+70}, {x+20, y+70}, {x+20, y+60}, {x+70, y+60},
                {x+70, y+30}, {x+20, y+30}, {x+20, y+20}, {x+100, y+20}};

set_color_nombre(el_GC,"Khaki");           llenaRectangulo(v, el_GC, x, y, 80, 55);
set_color_nombre(el_GC,"Magenta");         llenaRectangulo(v, el_GC, x, y+55, 80, 55);
set_color_nombre(el_GC,"Black");           pintar_flecha_arr(v, x-10, y+55, 55);
set_color_nombre(el_GC,"LightBlue");       llenaPoligono(v, el_GC, ppc, 20, 0, 0);
set_color_nombre(el_GC,"Black");           dibujaLineas(v, el_GC, ppc, 20, 0);
pintar_flecha_izq(v, x+110, y+15, 20);     pintar_flecha_izq(v, x-50, y+95, 20);

fontStruct=initFont(el_GC,"variable");
drawString(v, el_GC, x+110, y+5, "Wr Tre");
drawString(v, el_GC, x-75, y+85, "Wr Trs");
drawImageString(v, el_GC, x-25, y+80, "ncd");
drawString(v, el_GC, x+90, y+50, "Condensador");
drawString(v, el_GC, x+90, y+70, "Pcd  Tcd");

XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void pintar_bomba(Window v, int x, int y)
{
set_color_nombre(el_GC,"Black");           dibujaRectangulo(v, el_GC, x-1, y-21, 32, 12);
dibujaRectangulo(v, el_GC, x-31, y+9, 32, 12);   dibujaOvalo(v, el_GC, x-21, y-21, 42, 42);
set_color_nombre(el_GC,"Grey");            llenaRectangulo(v, el_GC, x, y-20, 30, 10);
llenaRectangulo(v, el_GC, x-30, y+10, 30, 10);   llenaOvalo(v, el_GC, x-20, y-20, 40, 40);
set_color_nombre(el_GC,"Black");

fontStruct=initFont(el_GC,"variable");
drawString(v, el_GC, x-15, y-30, "Bomba");
drawString(v, el_GC, x-10, y+35, "Wb");
drawString(v, el_GC, x-60, y+5, "Tisb");
XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void pintar_fuente(Window v, int x, int y)
{
set_color_nombre(el_GC,"Brown");
llenaRectangulo(v, el_GC, x, y, 20, 20);
set_color_nombre(el_GC,"Black");

fontStruct=initFont(el_GC,"variable");
drawString(v, el_GC, x, y-15, "Fuente");

XFlush(el_display);
XFreeFont(el_display, fontStruct);
}
/*=========================================*/
void pintar_flecha_arr(Window v, int x, int y, int l)
{
XPoint ppcf[3]={{x, y}, {x+4, y+8}, {x-4, y+8}};

set_color_nombre(el_GC,"Black");
dibujaLinea(v, el_GC, x, y, x, y+l);
llenaPoligono(v, el_GC, ppcf, 3, 0, 0);
XFlush(el_display);
}
```

```
/*=========================================*/
void pintar_flecha_abj(Window v, int x, int y, int l)
{
XPoint ppcf[3]={{x, y}, {x-4, y-8}, {x+4, y-8}};

set_color_nombre(el_GC,"Black");
llenaPoligono(v, el_GC, ppcf, 3, 0, 0);
dibujaLinea(v, el_GC, x, y, x, y-l);
XFlush(el_display);
}
/*=========================================*/
void pintar_flecha_izq(Window v, int x, int y, int l)
{
XPoint ppcf[3]={{x, y}, {x+8, y-4}, {x+8, y+4}};

set_color_nombre(el_GC,"Black");
llenaPoligono(v, el_GC, ppcf, 3, 0, 0);
dibujaLinea(v, el_GC, x, y, x+l, y);
XFlush(el_display);
}
/*=========================================*/
void pintar_flecha_der(Window v, int x, int y, int l)
{
XPoint ppcf[3]={{x, y}, {x-8, y+4}, {x-8, y-4}};

set_color_nombre(el_GC,"Black");
llenaPoligono(v, el_GC, ppcf, 3, 0, 0);
dibujaLinea(v, el_GC, x, y, x-l, y);
XFlush(el_display);
}
```

drawx.c

```
/* drawx.c */    /* INICIALIZACION DE LOS DIBUJOS */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>

#include "windows.h"
```

```
/*=========================================*/
#define CIRCULO_TOTAL (360*64)
#define EMPIEZA_CIRCULO 0

extern Display   *el_display;
/*=========================================*/
void dibujaLinea(Window ventana, GC el_GC, int x1, int y1, int x2, int y2)
    {
    XDrawLine(el_display, ventana, el_GC, x1, y1, x2, y2);
    }
/*=========================================*/
void dibujaPunto(Window ventana, GC el_GC, int x, int y)
    {
    XDrawPoint(el_display, ventana, el_GC, x, y);
    }
/*=========================================*/
void dibujaRectangulo(Window ventana, GC el_GC, int x, int y, int ancho, int alto)
    {
    XDrawRectangle(el_display, ventana, el_GC, x, y, ancho, alto);
    }
/*=========================================*/
void llenaRectangulo(Window ventana, GC el_GC, int x, int y, int ancho, int alto)
    {
    XFillRectangle(el_display, ventana, el_GC, x, y, ancho, alto);
    }
/*=========================================*/
void dibujaOvalo(Window ventana, GC el_GC, int x, int y, int ancho, int alto)
    {
    XDrawArc( el_display, ventana, el_GC, x, y, ancho, alto, EMPIEZA_CIRCULO, CIRCULO_TOTAL);
    }
/*=========================================*/
void llenaOvalo(Window ventana, GC el_GC, int x, int y, int ancho, int alto)
    {
    XFillArc( el_display, ventana, el_GC, x, y, ancho, alto, EMPIEZA_CIRCULO, CIRCULO_TOTAL);
    }
/*=========================================*/
void dibujaArco(Window ventana, GC el_GC, int x, int y, int ancho, int alto, int comienzo_angulo,
                int angulo)
    {
    XDrawArc( el_display, ventana, el_GC, x, y, ancho, alto, comienzo_angulo, angulo);
    }
/*=========================================*/
void llenaArco(Window ventana, GC el_GC, int x, int y, int ancho, int alto,
                int comienzo_angulo, int angulo)  /*no.grados hexagesimales x 64*/
```

```
    {
    XFillArc( el_display, ventana, el_GC, x, y, ancho, alto, comienzo_angulo, angulo);
    }
/*=============================================*/
void dibujaPuntos(Window ventana, GC el_GC, XPoint *losPuntos, int elNumeroDePuntos, int elModo)
    {
    XDrawPoints( el_display, ventana, el_GC, losPuntos, elNumeroDePuntos, elModo);
    }
/*=============================================*/
void dibujaLineas(Window ventana, GC el_GC, XPoint *losPuntos, int elNumeroDePuntos, int elModo)
    {
    XDrawLines( el_display, ventana, el_GC, losPuntos, elNumeroDePuntos, elModo);
    }
/*=============================================*/
void dibujaArcos(Window ventana, GC el_GC, XArc *losArcos, int elNumeroDeArcos)
    {
    XDrawArcs(el_display, ventana, el_GC, losArcos, elNumeroDeArcos);
    }
/*=============================================*/
void llenaArcos(Window ventana, GC el_GC, XArc *losArcos, int elNumeroDeArcos)
    {
    XFillArcs(el_display, ventana, el_GC, losArcos, elNumeroDeArcos);
    }
/*=============================================*/
void dibujaRectangulos(Window ventana, GC el_GC, XRectangle *lasRectas, int elNumeroDeRectas)
    {
    XDrawRectangles(el_display, ventana, el_GC, lasRectas, elNumeroDeRectas);
    }
/*=============================================*/
void llenaRectangulos(Window ventana, GC el_GC, XRectangle *lasRectas, int elNumeroDeRectas)
    {
    XFillRectangles(el_display, ventana, el_GC, lasRectas, elNumeroDeRectas);
    }
/*=============================================*/
void dibujaSegmentos(Window ventana, GC el_GC, XSegment segmentos[], int nsegmentos)
    {
    XDrawSegments(el_display, ventana, el_GC, segmentos, nsegmentos);
    }
/*=============================================*/
void llenaPoligono(Window ventana, GC el_GC, XPoint *puntos, int npuntos, int textura, int modo)
    {
    XFillPolygon(el_display, ventana, el_GC, puntos, npuntos, textura, modo);
    }
```

esquema1.c

```
/* esquema1.c */
/*=============================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
#include <stdlib.h>

#include "clases.h"
#include "windows.h"
/*=============================================*/
extern Window       ventana_raiz;
extern GC           el_GC;
extern Display      *el_display;

extern Window       wcn, wcn2;
extern Window       wd11, wd12, wd13, wd14, wd15, wd16;
extern Window       wcnt, wcnt00, wcnt01, wcnt10, wcnt11, wcnt20, wcnt21, wcnt30, wcnt31;
/*=============================================*/
void Scheduler::esquema_1(void)
{
/*---------------------------------------------*/
double  cont;
int     op;
XEvent Event;
/*---------------------------------------------*/
double u1d1, v1d1, u1d2, v1d2, u1d3, v1d3, u1d4, v1d4, u1d5, v1d5, u1d6, v1d6;
/*---------------------------------------------*/
Caldera cal(0.2, 1., 0.5, P_caldera);
Quemador quemador_1(1.0);
Fuente_de_liquido fuente(80.);
Valvula valvula_1(0.005);
Valvula valvula_2(0.005);
/*---------------------------------------------*/
/* Ojo, si cambiamos aqui,  tambien hacerlo en "abrir_ventanas_esquemax" y en  "esquema_mostrar" */
u1d1 = u1d2 = u1d3 = u1d4 = u1d5 = u1d6 = tf;
v1d1=cal.dar_l.c();  v1d2=22090.;        v1d3=374.5;
v1d4=200.;           v1d5=2000.;     v1d6=0.1;
/*---------------------------------------------*/
if(o5==1)
    {
    if((fp=fopen(filename,"a"))==NULL)
```

```
        { printf("Fichero NO puede abrirse"); exit(1);}
    fprintf(fp, "\nDatos_caract.:Lc\n");
    fprintf(fp, "%-7.4f", cal.dar_Lc());
    fprintf(fp, "\nDatos:t, nc, Pc, tc, dlc, hlc, slc, mlc, dvc, hvc, svc, mvc, Wlec, Wvsc, Qec, hlec.\n");
    }
/*-------------------------------------------------------------------*/
abrir_controles_esquema1();
imprimo_controles_esquema1(quemador_1.dar_Q(),valvula_2.dar_W(),
                                    valvula_1.dar_W(),fuente.dar_h());
abrir_ventanas_esquema1(tf);
/*-------------------------------------------------------------------*/
for(cont=hit, t=t0; t<tf; t=t+h, cont++)      /*simulacion esquema_1*/
    {
    cal.set_variables(quemador_1.dar_Q(), valvula_2.dar_W(), fuente.dar_h(), valvula_1.dar_W());
    cal.calculos_t();
/*-------------------------------------------------------------------*/
    if(o5==1)
        {
        if(cont==hit)
            {
            fprintf(fp, "%-7.1f ", t);
            fprintf(fp, "%-6.3f ", cal.dar_nc());
            fprintf(fp, "%-8.2f ", cal.dar_Pc());
            fprintf(fp, "%-6.2f ", cal.dar_tc());
            fprintf(fp, "%-10.5f ", cal.dar_dlc());
            fprintf(fp, "%-7.2f ", cal.dar_hlc());
            fprintf(fp, "%-6.4f ", cal.dar_slc());
            fprintf(fp, "%-8.4f ", cal.dar_mlc());
            fprintf(fp, "%-10.5f ", cal.dar_dvc());
            fprintf(fp, "%-7.2f ", cal.dar_hvc());
            fprintf(fp, "%-6.4f ", cal.dar_svc());
            fprintf(fp, "%-8.4f ", cal.dar_mvc());
            fprintf(fp, "%-8.5f ", valvula_2.dar_W());
            fprintf(fp, "%-8.5f ", valvula_1.dar_W());
            fprintf(fp, "%-5.1f ", quemador_1.dar_Q());
            fprintf(fp, "%-7.2f ", fuente.dar_h());
            cont=0.;
            }
        }
/*-------------------------------------------------------------------*/
    cambio_coordenadas(t, cal.dar_nc(), u1d1, v1d1);
    dibujo_punto(wd11, x, y, "Black");
    cambio_coordenadas(t, cal.dar_Pc(), u1d2, v1d2);
    dibujo_punto(wd12, x, y, "Yellow");
    cambio_coordenadas(t, cal.dar_tc(), u1d3, v1d3);
    dibujo_punto(wd13, x, y, "Black");
    cambio_coordenadas(t, quemador_1.dar_Q(), u1d4, v1d4);
    dibujo_punto(wd14, x, y, "Black");
    cambio_coordenadas(t, fuente.dar_h(), u1d5, v1d5);
    dibujo_punto(wd15, x, y, "Black");
    cambio_coordenadas(t, valvula_2.dar_W(), u1d6, v1d6);
    dibujo_punto(wd16, x, y, "White");
    cambio_coordenadas(t, valvula_1.dar_W(), u1d6, v1d6);
    dibujo_punto(wd16, x, y, "Black");
/*-------------------------------------------------------------------*/
    if(XCheckWindowEvent(el_display, wcn2, EV_MASK, &Event)==True)
        {
        do  {
            XNextEvent(el_display, &Event);
            switch(Event.type)
                {
                case ButtonPress:       op=raton_seguir(&Event.xbutton);break;
                default:        op=2;       break;
                }
            if(op==1) t=tf;
            }
            while(op==2);
        }
/*-------------------------------------------------------------------*/
    if(XCheckWindowEvent(el_display, wcnt00, EV_MASK, &Event)==True)
        {
        quemador_1.menos_Q();
        imprimo_controles_esquema1(quemador_1.dar_Q(), valvula_2.dar_W(), valvula_1.dar_W(),
                                                    fuente.dar_h());
        }
    if(XCheckWindowEvent(el_display, wcnt01, EV_MASK, &Event)==True)
        {
        quemador_1.mas_Q();
        imprimo_controles_esquema1(quemador_1.dar_Q(), valvula_2.dar_W(), valvula_1.dar_W(),
                                                    fuente.dar_h());
        }
    if(XCheckWindowEvent(el_display, wcnt10, EV_MASK, &Event)==True)
        {
        valvula_2.menos_W();
        imprimo_controles_esquema1(quemador_1.dar_Q(), valvula_2.dar_W(), valvula_1.dar_W(),
                                                    fuente.dar_h());
        }
    if(XCheckWindowEvent(el_display, wcnt11, EV_MASK, &Event)==True)
```

```
        {
        valvula_2.mas_W();
        imprimo_controles_esquema1(quemador_1.dar_Q(),valvula_2.dar_W(),valvula_1.dar_W(),
                                                    fuente.dar_h());
        }
    if(XCheckWindowEvent(el_display, wcnt20, EV_MASK, &Event)==True)
        {
        valvula_1.menos_W();
        imprimo_controles_esquema1(quemador_1.dar_Q(),valvula_2.dar_W(),valvula_1.dar_W(),
                                                    fuente.dar_h());
        }
    if(XCheckWindowEvent(el_display, wcnt21, EV_MASK, &Event)==True)
        {
        valvula_1.mas_W();
        imprimo_controles_esquema1(quemador_1.dar_Q(),valvula_2.dar_W(),valvula_1.dar_W(),
                                                    fuente.dar_h());
        }
    if(XCheckWindowEvent(el_display, wcnt30, EV_MASK, &Event)==True)
        {
        fuente.menos_h();
        imprimo_controles_esquema1(quemador_1.dar_Q(),valvula_2.dar_W(),valvula_1.dar_W(),
                                                    fuente.dar_h());
        }
    if(XCheckWindowEvent(el_display, wcnt31, EV_MASK, &Event)==True)
        {
        fuente.mas_h();
        imprimo_controles_esquema1(quemador_1.dar_Q(),valvula_2.dar_W(),valvula_1.dar_W(),
                                                    fuente.dar_h());
        }
/*-------------------------------------------------------------*/
    cal.inicializo_ncyPc(h);
/*-------------------------------------------------------------*/
    if (cal.dar_nc()>=0.99)         quemador_1.set_Q(0.);
    }
/*-------------------------------------------------------------*/
if(o5==1)
    fclose(fp);
/*-------------------------------------------------------------*/
cierro_controles();
do  {
    XNextEvent(el_display, &Event);
    switch(Event.type)
        {
        case ButtonPress:       op=raton_seguir(&Event.xbutton);break;
        default:                op=0;                   break;
        }
    }
    while(op!=1);
cierro_ventanas_esquema1();
}
/*=============================================*/
void abrir_controles_esquema1(void)
{
wcnt=abroVentana(10, 160, 330, 130, VENTANA_POP_UP, "wcnt", ESTADO_NORMAL, wcn,
&el_GC);
inicia_eventos(wcnt);

wcnt00=abroVentana(10, 10, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt00);
wcnt01=abroVentana(35, 10, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt01);
wcnt10=abroVentana(10, 40, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt10);
wcnt11=abroVentana(35, 40, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt11);
wcnt20=abroVentana(10, 70, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt20);
wcnt21=abroVentana(35, 70, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt21);
wcnt30=abroVentana(10, 100, 20, 20, 1, "", 0, wcnt, &el_GC);  inicia_eventos(wcnt30);
wcnt31=abroVentana(35, 100, 20, 20, 1, "", 0, wcnt, &el_GC);  inicia_eventos(wcnt31);

boton3(wcnt00, "LightBlue", "Yellow");      boton4(wcnt01, "LightBlue", "Yellow");
boton3(wcnt10, "LightBlue", "Yellow");      boton4(wcnt11, "LightBlue", "Yellow");
boton3(wcnt20, "LightBlue", "Yellow");      boton4(wcnt21, "LightBlue", "Yellow");
boton3(wcnt30, "LightBlue", "Yellow");      boton4(wcnt31, "LightBlue", "Yellow");
}
/*=============================================*/
void cierro_controles(void)
{
XDestroySubwindows(el_display,wcnt);
XDestroyWindow(el_display,wcnt);
XFlush(el_display);
}
/*=============================================*/
void imprimo_controles_esquema1(double Qec, double Wlec, double Wvsc, double hlec)
{
char buffer[50];

XClearArea(el_display, wcnt, 60, 10, 105, 110, 0);
XClearArea(el_display, wcnt, 220, 10, 105, 110, 0);

sprintf(buffer, "Qec=%-5.1f", Qec);       drawString(wcnt, el_GC, 70, 25, buffer);
sprintf(buffer, "Wlec=%-7.5f", Wlec);     drawString(wcnt, el_GC, 70, 55, buffer);
```

```
    sprintf(buffer, "Wvsc=%-7.5f", Wvsc);          drawString(wcnt, el_GC, 70, 85, buffer);
    sprintf(buffer, "hlec=%-9.4f", hlec);          drawString(wcnt, el_GC, 70, 115, buffer);
    }
    /*=========================================*/
    void abrir_ventanas_esquema1(double tf)
    {
    wd11=abroVentana(5, 390, 280, 250, VENTANA_POP_UP, "wd1", 0, ventana_raiz, &el_GC);
    wd12=abroVentana(290, 390, 280, 250, VENTANA_POP_UP, "wd2", 0, ventana_raiz, &el_GC);
    wd13=abroVentana(575, 390, 280, 250, VENTANA_POP_UP, "wd3", 0, ventana_raiz, &el_GC);
    wd14=abroVentana(5, 645, 280, 250, VENTANA_POP_UP, "wd4", 0, ventana_raiz, &el_GC);
    wd15=abroVentana(290, 645, 280, 250, VENTANA_POP_UP, "wd5", 0, ventana_raiz, &el_GC);
    wd16=abroVentana(575, 645, 280, 250, VENTANA_POP_UP, "wd6", 0, ventana_raiz, &el_GC);

    dibujo_entorno(wd11, "Nivel Caldera (m)", "SkyBlue", 1, "nc", "", "", "", "Black", "", "", "", 1., tf);
    dibujo_entorno(wd12, "Presion Caldera (kPa)", "SkyBlue", 1, "Pc", "", "", "", "Yellow", "", "", "",
22090., tf);
    dibujo_entorno(wd13, "Temperatura Caldera (C)", "SkyBlue", 1, "Tc", "", "", "", "Black", "", "", "",
374.5, tf);
    dibujo_entorno(wd14, "Flujo de Calor (kJ/s)", "MediumAquamarine", 1, "Qec", "", "", "", "Black", "", "",
"",
            200., tf);
    dibujo_entorno(wd15, "Entalpia de entrada (kJ/kg)", "MediumAquamarine", 1, "hlec", "", "", "", "Black",
            "", "", "", 2000., tf);
    dibujo_entorno(wd16, "Caudal (kg/s)", "MediumAquamarine", 2, "Wlec", "Wvsc", "", "", "White",
"Black",
            "", "", 0.1, tf);
    }
    /*=========================================*/
    voidcierro_ventanas_esquema1(void)
    {
    XDestroyWindow(el_display, wd11);
    XDestroyWindow(el_display, wd12);
    XDestroyWindow(el_display, wd13);
    XDestroyWindow(el_display, wd14);
    XDestroyWindow(el_display, wd15);
    XDestroyWindow(el_display, wd16);
    XFlush(el_display);
    }
```

esquema2.c

```
/* esquema2.c */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
#include <stdlib.h>

#include "clases.h"
#include "windows.h"
/*=========================================*/
extern Window  ventana_raiz;
extern GC      el_GC;
extern Display  *el_display;

extern Window  wcn, wcn2;
extern Window  wd21, wd22, wd23, wd24, wd25, wd26, wd27, wd28;
extern Window  wcnt, wcnt00, wcnt01, wcnt10, wcnt11, wcnt20, wcnt21, wcnt30, wcnt31;
/*=========================================*/
void Scheduler::esquema_2(void)
{
/*-----------------------------------------------------------------*/
double  cont;
int     op;
XEvent Event;
/*-----------------------------------------------------------------*/
double  uld1, vld1, uld2, vld2, uld3, vld3, uld4, vld4, uld5, vld5, uld6, vld6, uld7, vld7,
        uld8, vld8;
/*-----------------------------------------------------------------*/
Caldera cal(0.2, 1., 0.5, P_caldera);
Quemador quemador_1(1.0);
Quemador quemador_2(5.0);
Fuente_de_liquido fuente(80.);
Valvula valvula_1(0.005);
Valvula valvula_2(0.005);
Valvula valvula_3(valvula_2.dar_W());
Supercalentador sup(2850., 1., 10.);
/*-----------------------------------------------------------------*/
```

```
/* Ojo, si cambiamos aqui,  tambien hacerlo en "abrir_ventanas_esquemax" y en "esquema_mostrar" */
u1d1 = u1d2 = u1d3 = u1d4 = u1d5 = u1d6 = u1d7 = u1d8 = tf;
v1d1=cal.dar_Lc();  v1d2=50000.;        v1d3=926.87;        v1d4=4000.;
v1d5=1.;            v1d6=10.;           v1d7=100.;          v1d8=0.1;
/*-------------------------------------------------------------------*/
if(o5==1)
    {
    if((fp=fopen(filename,"a"))==NULL)
        { printf("Fichero NO puede abrirse"); exit(1);}
    fprintf(fp,"\nDatos_caract.:Lc\n");
    fprintf(fp, "%-7.4f", cal.dar_Lc());
    fprintf(fp, "\nDatos:t, nc, Pves, Pvss, Tves, Tvss, hves, hvss, hlec, vves, vvss, sves, svss,
            Qec, Qes, Wlec, Wvsc.\n");
    }
/*-------------------------------------------------------------------*/
abrir_controles_esquema2();
imprimo_controles_esquema2(quemador_1.dar_Q(),quemador_2.dar_Q(),valvula_2.dar_W(),
                    valvula_1.dar_W());
abrir_ventanas_esquema2(tf);
/*-------------------------------------------------------------------*/
for(cont=hit, t=t0; t<tf; t=t+h, cont++)      /*simulacion esquema_2*/
    {
    cal.set_variables(quemador_1.dar_Q(), valvula_2.dar_W(), fuente.dar_h(), valvula_1.dar_W());
                                                cal.calculos_t();
    sup.set_variables(cal.dar_Pc(), cal.dar_tc(), cal.dar_hvc(), 1./cal.dar_dvc(), quemador_2.dar_Q(),
                                            valvula_1.dar_W(), cal.dar_svc());
    sup.calculos_t(t,sup.dar_Wves());
    /*-------------------------------------------------------------*/
    if(o5==1)
        {
        if(cont==hit)
            {
            fprintf(fp, "%-7.1f ", t);
            fprintf(fp, "%-6.3f ", cal.dar_nc());
            fprintf(fp, "%-8.2f ", sup.dar_Pves());
            fprintf(fp, "%-8.2f ", sup.dar_Pvss());
            fprintf(fp, "%-6.2f ", sup.dar_Tves());
            fprintf(fp, "%-6.2f ", sup.dar_Tvss());
            fprintf(fp, "%-7.2f ", sup.dar_hves());
            fprintf(fp, "%-7.2f ", sup.dar_hvss());
            fprintf(fp, "%-7.2f ", fuente.dar_h());
            fprintf(fp, "%-9.6f ", sup.dar_vves());
            fprintf(fp, "%-9.6f ", sup.dar_vvss());
            fprintf(fp, "%-6.4f ", sup.dar_sves());
            fprintf(fp, "%-6.4f ", sup.dar_svss());
            fprintf(fp, "%-5.1f ", quemador_1.dar_Q());
            fprintf(fp, "%-5.1f ", quemador_2.dar_Q());
            fprintf(fp, "%-8.5f ", valvula_2.dar_W());
            fprintf(fp, "%-8.5f ", valvula_1.dar_W());
            cont=0.;
            }
        }
/*-------------------------------------------------------------------*/
cambio_coordenadas(t, cal.dar_nc(), u1d1, v1d1);
dibujo_punto(wd21, x, y, "Black");

cambio_coordenadas(t, sup.dar_Pves(), u1d2, v1d2);
dibujo_punto(wd22, x, y, "Black");
cambio_coordenadas(t, sup.dar_Pvss(), u1d2, v1d2);
dibujo_punto(wd22, x, y, "Yellow");

cambio_coordenadas(t, sup.dar_Tves(), u1d3, v1d3);
dibujo_punto(wd23, x, y, "Black");
cambio_coordenadas(t, sup.dar_Tvss(), u1d3, v1d3);
dibujo_punto(wd23, x, y, "Yellow");

cambio_coordenadas(t, sup.dar_hves(), u1d4, v1d4);
dibujo_punto(wd24, x, y, "Black");
cambio_coordenadas(t, sup.dar_hvss(), u1d4, v1d4);
dibujo_punto(wd24, x, y, "Yellow");
cambio_coordenadas(t, fuente.dar_h(), u1d4, v1d4);
dibujo_punto(wd24, x, y, "White");

cambio_coordenadas(t, sup.dar_vves(), u1d5, v1d5);
dibujo_punto(wd25, x, y, "Black");
cambio_coordenadas(t, sup.dar_vvss(), u1d5, v1d5);
dibujo_punto(wd25, x, y, "Yellow");

cambio_coordenadas(t, sup.dar_sves(), u1d6, v1d6);
dibujo_punto(wd26, x, y, "Black");
cambio_coordenadas(t, sup.dar_svss(), u1d6, v1d6);
dibujo_punto(wd26, x, y, "Yellow");

cambio_coordenadas(t, quemador_1.dar_Q(), u1d7, v1d7);
dibujo_punto(wd27, x, y, "White");
cambio_coordenadas(t, quemador_2.dar_Q(), u1d7, v1d7);
dibujo_punto(wd27, x, y, "Black");
```

```
        { valvula_1.menos_W();
        imprimo_controles_esquema2(quemador_1.dar_Q(),quemador_2.dar_Q(),
                                valvula_2.dar_W(), valvula_1.dar_W()); }
    if(XCheckWindowEvent(el_display, wcnt31, EV_MASK, &Event)==True)
        { valvula_1.mas_W();
        imprimo_controles_esquema2(quemador_1.dar_Q(),quemador_2.dar_Q(),
        valvula_2.dar_W(), valvula_1.dar_W()); }
    /*-----------------------------------------------------------------*/
    cal.inicializo_ncyPc(h);
    sup.inicializo_hvss(t+h);
    /*-----------------------------------------------------------------*/
    /* Aquí van los controles */
    if(sup.dar_Tvss()820.)quemador_2.set_Q(0.);
    }
/*-----------------------------------------------------------------*/
if(o5==1)
    fclose(fp);
/*-----------------------------------------------------------------*/
cierro_controles();
do  {
    XNextEvent(el_display, &Event);
    switch(Event.type)
        {
        case ButtonPress:   op=raton_seguir(&Event.xbutton);break;
        default:            op=0;                           break;
        }
    }
    while(op!=1);
cierro_ventanas_esquema2();
}
/*=================================================*/
void abrir_controles_esquema2(void)
{
wcnt=abroVentana(10, 160, 330, 130, VENTANA_POP_UP, "wcnt", ESTADO_NORML, wcn,
&el_GC);
inicia_eventos(wcnt);

wcnt00=abroVentana(10, 10, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt00);
wcnt01=abroVentana(35, 10, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt01);
wcnt10=abroVentana(10, 40, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt10);
wcnt11=abroVentana(35, 40, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt11);
wcnt20=abroVentana(10, 70, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt20);
wcnt21=abroVentana(35, 70, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt21);
wcnt30=abroVentana(10, 100, 20, 20, 1, "", 0, wcnt, &el_GC);  inicia_eventos(wcnt30);
```



```
cambio_coordenadas(t, valvula_2.dar_W(), u1d8, v1d8);
dibujo_punto(wd28, x, y, "White");
cambio_coordenadas(t, valvula_1.dar_W(), u1d8, v1d8);
dibujo_punto(wd28, x, y, "Black");
/*-----------------------------------------------------------------*/
if(XCheckWindowEvent(el_display, wcn2, EV_MASK, &Event)==True)
    {
    do  {
        XNextEvent(el_display, &Event);
        switch(Event.type)
            {
            case ButtonPress:   op=raton_seguir(&Event.xbutton);break;
            default:            op=2;                           break;
            }
        if(op==1) t=tf;
        }
        while(op==2);
    }
/*-----------------------------------------------------------------*/
if(XCheckWindowEvent(el_display, wcnt00, EV_MASK, &Event)==True)
    { quemador_1.menos_Q();
    imprimo_controles_esquema2(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        valvula_2.dar_W(), valvula_1.dar_W()); }
if(XCheckWindowEvent(el_display, wcnt01, EV_MASK, &Event)==True)
    { quemador_1.mas_Q();
    imprimo_controles_esquema2(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        valvula_2.dar_W(), valvula_1.dar_W()); }
if(XCheckWindowEvent(el_display, wcnt10, EV_MASK, &Event)==True)
    { quemador_2.menos_Q();
    imprimo_controles_esquema2(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        valvula_2.dar_W(), valvula_1.dar_W()); }
if(XCheckWindowEvent(el_display, wcnt11, EV_MASK, &Event)==True)
    { quemador_2.mas_Q();
    imprimo_controles_esquema2(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        valvula_2.dar_W(), valvula_1.dar_W()); }
if(XCheckWindowEvent(el_display, wcnt20, EV_MASK, &Event)==True)
    { valvula_2.menos_W();
    imprimo_controles_esquema2(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        valvula_2.dar_W(), valvula_1.dar_W()); }
if(XCheckWindowEvent(el_display, wcnt21, EV_MASK, &Event)==True)
    { valvula_2.mas_W();
    imprimo_controles_esquema2(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        valvula_2.dar_W(), valvula_1.dar_W()); }
if(XCheckWindowEvent(el_display, wcnt30, EV_MASK, &Event)==True)
```

```
    wcnt31=abroVentana(35, 100, 20, 20, 1, "", 0, wcnt, &el_GC); inicia_eventos(wcnt31);

    boton3(wcnt00, "LightBlue", "Yellow");      boton4(wcnt01, "LightBlue", "Yellow");
    boton3(wcnt10, "LightBlue", "Yellow");      boton4(wcnt11, "LightBlue", "Yellow");
    boton3(wcnt20, "LightBlue", "Yellow");      boton4(wcnt21, "LightBlue", "Yellow");
    boton3(wcnt30, "LightBlue", "Yellow");      boton4(wcnt31, "LightBlue", "Yellow");
    }
    /*=========================================*/
    void imprimo_controles_esquema2(double Qec, double Qes, double Wlec, double Wvsc)
    {
    char buffer[50];

    XClearArea(el_display, wcnt, 60, 10, 105, 110, 0);
    XClearArea(el_display, wcnt, 220, 10, 105, 110, 0);

    sprintf(buffer, "Qec=%-5.1f", Qec);      drawString(wcnt, el_GC, 70, 25, buffer);
    sprintf(buffer, "Qes=%-5.1f", Qes);      drawString(wcnt, el_GC, 70, 55, buffer);
    sprintf(buffer, "Wlec=%-7.5f", Wlec);    drawString(wcnt, el_GC, 70, 85, buffer);
    sprintf(buffer, "Wvsc=%-7.5f", Wvsc);    drawString(wcnt, el_GC, 70, 115, buffer);
    }
    /*=========================================*/
    void abrir_ventanas_esquema2(double tf)
    {
    wd21=abroVentana(5, 390, 280, 250, VENTANA_POP_UP, "wd21", 0, ventana_raiz, &el_GC);
    wd22=abroVentana(290, 390, 280, 250, VENTANA_POP_UP, "wd22", 0, ventana_raiz, &el_GC);
    wd23=abroVentana(575, 390, 280, 250, VENTANA_POP_UP, "wd23", 0, ventana_raiz, &el_GC);
    wd24=abroVentana(860, 390, 280, 250, VENTANA_POP_UP, "wd24", 0, ventana_raiz, &el_GC);

    wd25=abroVentana(5, 645, 280, 250, VENTANA_POP_UP, "wd25", 0, ventana_raiz, &el_GC);
    wd26=abroVentana(290, 645, 280, 250, VENTANA_POP_UP, "wd26", 0, ventana_raiz, &el_GC);
    wd27=abroVentana(575, 645, 280, 250, VENTANA_POP_UP, "wd27", 0, ventana_raiz, &el_GC);
    wd28=abroVentana(860, 645, 280, 250, VENTANA_POP_UP, "wd28", 0, ventana_raiz, &el_GC);

    dibujo_entorno(wd21, "Nivel Caldera (m)", "SkyBlue", 1, "nc", "", "", "", "Black", "", "", "", 1., tf);
    dibujo_entorno(wd22, "Presion (kPa)", "SkyBlue", 2, "Pves", "Pvss", "", "","Black","Yellow", "",
"",50000., tf);
    dibujo_entorno(wd23, "Tempratura (C)", "SkyBlue", 2,"Tves","Tvss","", "","Black","Yellow","", "",
926.87,tf);
    dibujo_entorno(wd24, "Entalpia (kJ/kg)", "SkyBlue", 3, "hves", "hvss", "hlec", "", "Black", "Yellow",
"White",
                   "", 4000., tf);
    dibujo_entorno(wd25, "Volumen especifico (m3/kg)", "SkyBlue", 2, "vves", "vvss", "", "", "Black", "Yel-
low",
                   "", "", 1., tf);
    dibujo_entorno(wd26, "Etropia (kJ/(kg.K))", "SkyBlue",2,"sves","svss", "", "","Black","Yellow","", "",
10., tf);
    dibujo_entorno(wd27, "Flujo de Calor (kJ/s)", "MediumAquamarine", 2, "Qec", "Qes", "", "", "White",
                                                    "Black", "", "", 100., tf);
    dibujo_entorno(wd28, "Caudal (kg/s)", "MediumAquamarine", 2, "Wlec", "Wvsc", "", "", "White",
"Black",
                                                    "", "", 0.1, tf);
    }
    /*=========================================*/
    voidcierro_ventanas_esquema2(void)
    {
    XDestroyWindow(el_display, wd21);
    XDestroyWindow(el_display, wd22);
    XDestroyWindow(el_display, wd23);
    XDestroyWindow(el_display, wd24);
    XDestroyWindow(el_display, wd25);
    XDestroyWindow(el_display, wd26);
    XDestroyWindow(el_display, wd27);
    XDestroyWindow(el_display, wd28);
    XFlush(el_display);
    }
```

esquema3.c

```
/* esquema3.c */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
#include <stdlib.h>
#include<math.h>

#include "clases.h"
#include "windows.h"
/*=========================================*/
extern Window       ventana_raiz;
extern GC           el_GC;
extern Display      *el_display;
```

```
extern Window       wcn, wcn2;
extern Window       wd31, wd32, wd33, wd34, wd35, wd36, wd37, wd38, wd39, wd310;
extern Window       wcnt, wcnt00, wcnt01, wcnt10, wcnt11, wcnt20, wcnt21, wcnt30, wcnt31;
extern Window       wcnt40, wcnt41, wcnt50, wcnt51, wcnt60, wcnt61, wcnt70, wcnt71;
/*=============================================*/
void Scheduler::esquema_3(void)
{
/*---------------------------------------------------------------------*/
double  cont, R;
int     op;
XEvent Event;
/*---------------------------------------------------------------------*/
double  u1d1, v1d1, u1d2, v1d2, u1d3, v1d3, u1d4, v1d4, u1d5, v1d5,
        u1d6, v1d6, u1d7, v1d7, u1d8, v1d8, u1d9, v1d9, u1d10, v1d10;
/*---------------------------------------------------------------------*/
Caldera cal(0.2, 1., 0.5, P_caldera);
Quemador quemador_1(1.0);
Valvula valvula_1(W_valvula1);
//Supercalentador sup(2850., 1., 10.);
Supercalentador sup(h_supercal, m_supercal, k_supercal);
Quemador quemador_2(1.0);
Turbina turb(kx_turbina);
Condensador cond(0.2,1.0, 0.5, P_condensador, W_refrigerante, m_refrigerante,
                                T_refrig_in, T_refrig_out);
Bomba bomb(W_bomba);
/*---------------------------------------------------------------------*/
/* Ojo, si cambiamos aqui,  tambien hacerlo en "abrir_ventanas_esquemax" y en "esquema_mostrar" */
u1d1 = u1d2 = u1d3 = u1d4 = u1d5 = u1d6 = u1d7 = u1d8 = u1d9 = u1d10 = tf;
v1d1=cal.dar_Lc();  v1d2=30000.;   v1d3=926.87;   v1d4=4000.;v1d5=2.;
v1d6=926.87;        v1d7=100.;     v1d8=0.1;      v1d9=100.;v1d10=1.;
/*---------------------------------------------------------------------*/
if(o5==1)
    {
    if((fp=fopen(filename, "a"))==NULL)
        { printf("Fichero NO puede abrirse"); exit(1);}
    fprintf(fp, "\nDatos_caract.:Lc\n");
    fprintf(fp, "%-7.4f", cal.dar_Lc());
    fprintf(fp, "\nDatos:t, nc, ncd, Pc, Pcd, Tc, Tcd, hves, hvss, hvst, vves, vvss, vvst, Tvss, Tvst, Tlsb,
                                Qec, Qes, Wvst, Wvsc, Wb, Wr, Tre, Trs, kx, R.\n");
    }
/*---------------------------------------------------------------------*/
abrir_controles_esquema3();
imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),cond.dar_Tre(),
                    cond.dar_Trs(), valvula_1.dar_W(), bomb.dar_Wb(), cond.dar_Wr(),
                    turb.dar_kx());
abrir_ventanas_esquema3(tf);
/*---------------------------------------------------------------------*/
for(cont=hit, t=t0; t<tf; t=t+h, cont++)      /*simulacion esquema_3*/
    {
    cond.set1_variables();
    bomb.set_variables(cond.dar_Pcd(), cond.dar_dlcd(), cal.dar_Pc());
    cal.set_variables(quemador_1.dar_Q(), bomb.dar_Wb(), bomb.dar_hlsb(), valvula_1.dar_W());
    cal.calculos_t();
    sup.set_variables(cal.dar_Pc(), cal.dar_tc(), cal.dar_hvc(), 1./cal.dar_dvc(), quemador_2.dar_Q(),
                                valvula_1.dar_W(), cal.dar_svc());
    sup.calculos_t(t, sup.dar_Wves());
    turb.set_variables(sup.dar_Wvss(), sup.dar_Pvss(), sup.dar_Tvss(),
    sup.dar_hvss(), sup.dar_svss(), sup.dar_vvss(), cond.dar_Pcd());
    cond.set2_variables(turb.dar_Wvst(), turb.dar_Tvst(), turb.dar_xvst(), turb.dar_hvst(),
                                turb.dar_vvst(), bomb.dar_Wb());

    R=(sup.dar_hvss()+cond.dar_hlcd()-turb.dar_hvst()-bomb.dar_hlsb())/
                    fabs(cal.dar_hlc()+sup.dar_hvss()+(cal.dar_tc()+273.)*
                    (sup.dar_sves()-cal.dar_slc())-bomb.dar_hlsb()-sup.dar_hves());
    /*---------------------------------------------------------------------*/
    if(o5==1)
        {
        if(cont==hit)
            {
            fprintf(fp, "%-7.1f", t);
            fprintf(fp, "%-6.3f", cal.dar_nc());
            fprintf(fp, "%-6.3f", cond.dar_ncd());
            fprintf(fp, "%-8.2f", cal.dar_Pc());
            fprintf(fp, "%-8.2f", cond.dar_Pcd());
            fprintf(fp, "%-6.2f", cal.dar_tc());
            fprintf(fp, "%-6.2f", cond.dar_Tcd());
            fprintf(fp, "%-7.2f", sup.dar_hves());
            fprintf(fp, "%-7.2f", sup.dar_hvss());
            fprintf(fp, "%-7.2f", turb.dar_hvst());
            fprintf(fp, "%-9.5f", sup.dar_vves());
            fprintf(fp, "%-9.5f", sup.dar_vvss());
            fpintf(fp, "%-9.5f", turb.dar_vvst());
            fprintf(fp, "%-6.2f", sup.dar_Tvss());
            fprintf(fp, "%-6.2f", turb.dar_Tvst());
            fprintf(fp, "%-6.2f", bomb.dar_Tlsb());
            fprintf(fp, "%-5.1f", quemador_1.dar_Q());
            fprintf(fp, "%-5.1f", quemador_2.dar_Q());
            fprintf(fp, "%-8.5f", turb.dar_Wvst());
```

```
        fprintf(fp, "%-8.5f ", valvula_1.dar_W());
        fprintf(fp, "%-8.5f ", bomb.dar_Wb());
        fprintf(fp, "%-8.5f ", cond.dar_Wr());
        fprintf(fp, "%-6.2f ", cond.dar_Tre());
        fprintf(fp, "%-6.2f ", cond.dar_Trs());
        fprintf(fp, "%-4.2f ", turb.dar_kx());
        fprintf(fp, "%-4.2f ", R);
        cont=0.;
    }
}
/*-----------------------------------------------------------------*/
cambio_coordenadas(t, cal.dar_nc(), u1d1, v1d1);      dibujo_punto(wd31, x, y, "Black");
cambio_coordenadas(t, cond.dar_ncd(), u1d1, v1d1);    dibujo_punto(wd31, x, y, "Yellow");

cambio_coordenadas(t, cal.dar_Pc(), u1d2, v1d2);      dibujo_punto(wd32, x, y, "Black");
cambio_coordenadas(t, cond.dar_Pcd(), u1d2, v1d2);    dibujo_punto(wd32, x, y, "Yellow");

cambio_coordenadas(t, cal.dar_tc(), u1d3, v1d3);      dibujo_punto(wd33, x, y, "Black");
cambio_coordenadas(t, cond.dar_Tcd(), u1d3, v1d3);    dibujo_punto(wd33, x, y, "Yellow");

cambio_coordenadas(t, sup.dar_hves(), u1d4, v1d4);    dibujo_punto(wd34, x, y, "Black");
cambio_coordenadas(t, sup.dar_hvss(), u1d4, v1d4);    dibujo_punto(wd34, x, y, "White");
cambio_coordenadas(t, turb.dar_hvst(), u1d4, v1d4);   dibujo_punto(wd34, x, y, "Yellow");

cambio_coordenadas(t, sup.dar_vves(), u1d5, v1d5);    dibujo_punto(wd35, x, y, "Black");
cambio_coordenadas(t, sup.dar_vvss(), u1d5, v1d5);    dibujo_punto(wd35, x, y, "White");
cambio_coordenadas(t, turb.dar_vvst(), u1d5, v1d5);   dibujo_punto(wd35, x, y, "Yellow");

cambio_coordenadas(t, sup.dar_Tvss(), u1d6, v1d6);    dibujo_punto(wd36, x, y, "White");
cambio_coordenadas(t, turb.dar_Tvst(), u1d6, v1d6);   dibujo_punto(wd36, x, y, "Yellow");
cambio_coordenadas(t, bomb.dar_Tlsb(), u1d6, v1d6); dibujo_punto(wd36, x, y, "Black");

cambio_coordenadas(t, quemador_1.dar_Q(), u1d7, v1d7); dibujo_punto(wd37, x, y, "Black");
cambio_coordenadas(t, quemador_2.dar_Q(), u1d7, v1d7); dibujo_punto(wd37, x, y, "White");

cambio_coordenadas(t, turb.dar_Wvst(), u1d8, v1d8);  dibujo_punto(wd38, x, y, "White");
cambio_coordenadas(t, valvula_1.dar_W(), u1d8, v1d8);dibujo_punto(wd38, x, y, "Black");
cambio_coordenadas(t, bomb.dar_Wb(), u1d8, v1d8);    dibujo_punto(wd38, x, y, "Red");
cambio_coordenadas(t, cond.dar_Wr(), u1d8, v1d8);    dibujo_punto(wd38, x, y, "Yellow");

cambio_coordenadas(t, cond.dar_Tre(), u1d9, v1d9);   dibujo_punto(wd39, x, y, "White");
cambio_coordenadas(t, cond.dar_Trs(), u1d9, v1d9);   dibujo_punto(wd39, x, y, "Black");

cambio_coordenadas(t, R, u1d10, v1d10);          dibujo_punto(wd310, x, y, "Blue");
cambio_coordenadas(t, turb.dar_kx(), u1d10, v1d10);   dibujo_punto(wd310, x, y, "Black");
/*-----------------------------------------------------------------*/
if(XCheckWindowEvent(el_display, wcn2, EV_MASK, &Event)==True)
    {
    do  {
        XNextEvent(el_display, &Event);
        switch(Event.type)
            {
            case ButtonPress:       op=raton_seguir(&Event.xbutton);break;
            default:        op=2;                   break;
            }
        if(op==1) t=tf;
        }
    while(op==2);
    }
/*-----------------------------------------------------------------*/
if(XCheckWindowEvent(el_display, wcnt00, EV_MASK, &Event)==True)
    { quemador_1.menos_Q();
    imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                    cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                    bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
if(XCheckWindowEvent(el_display, wcnt01, EV_MASK, &Event)==True)
    { quemador_1.mas_Q();
    imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                    cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                    bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
if(XCheckWindowEvent(el_display, wcnt10, EV_MASK, &Event)==True)
    { quemador_2.menos_Q();
    imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                    cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                    bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
if(XCheckWindowEvent(el_display, wcnt11, EV_MASK, &Event)==True)
    { quemador_2.mas_Q();
    imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                    cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                    bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
if(XCheckWindowEvent(el_display, wcnt20, EV_MASK, &Event)==True)
    { cond.menos_Tre();
    imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                    cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                    bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
if(XCheckWindowEvent(el_display, wcnt21, EV_MASK, &Event)==True)
    { cond.mas_Tre();
    imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
```

```
            imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
        if(XCheckWindowEvent(el_display, wcnt71, EV_MASK, &Event)==True)
            { turb.mas_kx();
            imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
        /*-------------------------------------------------------------*/
        cal.inicializo_ncyPc(h);
        sup.inicializo_hvss(t+h);
        cond.inicializo_ncdyPcd(h);
        /*-------------------------------------------------------------*/
        /* Aquí van las líneas de los controles */
        if(sup.dar_Tvss()820.)quemador_2.set_Q(0.);
        }
    /*-----------------------------------------------------------------*/
    if(o5==1)
        fclose(fp);
    /*-----------------------------------------------------------------*/
    cierro_controles();
    do  {
        XNextEvent(el_display, &Event);
        switch(Event.type)
            {
            case ButtonPress:       op=raton_seguir(&Event.xbutton);break;
            default:        op=0;                   break;
            }
        }
        while(op!=1);
    cierro_ventanas_esquema3();
    }
    /*=================================================*/
    voidabrir_controles_esquema3(void)
    {
    wcnt=abroVentana(10, 160, 330, 130, VENTANA_POP_UP, "wcnt", ESTADO_NORML, wcn,
&el_GC);
    inicia_eventos(wcnt);

    wcnt00=abroVentana(10, 10, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt00);
    wcnt01=abroVentana(35, 10, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt01);
    wcnt10=abroVentana(10, 40, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt10);
    wcnt11=abroVentana(35, 40, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt11);
    wcnt20=abroVentana(10, 70, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt20);
```



```
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
    if(XCheckWindowEvent(el_display, wcnt30, EV_MASK, &Event)==True)
        { cond.menos_Trs();
        imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
    if(XCheckWindowEvent(el_display, wcnt31, EV_MASK, &Event)==True)
        { cond.mas_Trs();
        imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
    if(XCheckWindowEvent(el_display, wcnt40, EV_MASK, &Event)==True)
        { valvula_1.menos_W();
        imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
    if(XCheckWindowEvent(el_display, wcnt41, EV_MASK, &Event)==True)
        { valvula_1.mas_W();
        imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
    if(XCheckWindowEvent(el_display, wcnt50, EV_MASK, &Event)==True)
        { bomb.menos_Wb();
        imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
    if(XCheckWindowEvent(el_display, wcnt51, EV_MASK, &Event)==True)
        { bomb.mas_Wb();
        imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
    if(XCheckWindowEvent(el_display, wcnt60, EV_MASK, &Event)==True)
        { cond.menos_Wr();
        imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
    if(XCheckWindowEvent(el_display, wcnt61, EV_MASK, &Event)==True)
        { cond.mas_Wr();
        imprimo_controles_esquema3(quemador_1.dar_Q(),quemador_2.dar_Q(),
                        cond.dar_Tre(), cond.dar_Trs(), valvula_1.dar_W(),
                        bomb.dar_Wb(), cond.dar_Wr(), turb.dar_kx()); }
    if(XCheckWindowEvent(el_display, wcnt70, EV_MASK, &Event)==True)
        { turb.menos_kx();
```

```
wcnt21=abroVentana(35, 70, 20, 20, 1, "", 0, wcnt, &el_GC);   inicia_eventos(wcnt21);
wcnt30=abroVentana(10, 100, 20, 20, 1, "", 0, wcnt, &el_GC); inicia_eventos(wcnt30);
wcnt31=abroVentana(35, 100, 20, 20, 1, "", 0, wcnt, &el_GC); inicia_eventos(wcnt31);

wcnt40=abroVentana(170, 10, 20, 20, 1, "", 0, wcnt, &el_GC); inicia_eventos(wcnt40);
wcnt41=abroVentana(195, 10, 20, 20, 1, "", 0, wcnt, &el_GC); inicia_eventos(wcnt41);
wcnt50=abroVentana(170, 40, 20, 20, 1, "", 0, wcnt, &el_GC); inicia_eventos(wcnt50);
wcnt51=abroVentana(195, 40, 20, 20, 1, "", 0, wcnt, &el_GC); inicia_eventos(wcnt51);
wcnt60=abroVentana(170, 70, 20, 20, 1, "", 0, wcnt, &el_GC); inicia_eventos(wcnt60);
wcnt61=abroVentana(195, 70, 20, 20, 1, "", 0, wcnt, &el_GC); inicia_eventos(wcnt61);
wcnt70=abroVentana(170, 100, 20, 20, 1, "", 0, wcnt, &el_GC);inicia_eventos(wcnt70);
wcnt71=abroVentana(195, 100, 20, 20, 1, "", 0, wcnt, &el_GC);inicia_eventos(wcnt71);
boton3(wcnt00, "LightBlue", "Yellow");       boton4(wcnt01, "LightBlue", "Yellow");
boton3(wcnt10, "LightBlue", "Yellow");       boton4(wcnt11, "LightBlue", "Yellow");
boton3(wcnt20, "LightBlue", "Yellow");       boton4(wcnt21, "LightBlue", "Yellow");
boton3(wcnt30, "LightBlue", "Yellow");       boton4(wcnt31, "LightBlue", "Yellow");

boton3(wcnt40, "LightBlue", "Yellow");       boton4(wcnt41, "LightBlue", "Yellow");
boton3(wcnt50, "LightBlue", "Yellow");       boton4(wcnt51, "LightBlue", "Yellow");
boton3(wcnt60, "LightBlue", "Yellow");       boton4(wcnt61, "LightBlue", "Yellow");
boton3(wcnt70, "LightBlue", "Yellow");       boton4(wcnt71, "LightBlue", "Yellow");
}
/*=========================================*/
void imprimo_controles_esquema3(double Qec, double Qes, double Tre, double Trs, double Wvsc,
                                double Wb, double Wr, double kx)
{
char buffer[50];

XClearArea(el_display, wcnt, 60, 10, 105, 110, 0);
XClearArea(el_display, wcnt, 220, 10, 105, 110, 0);

sprintf(buffer, "Qec=%-5.1f", Qec);       drawString(wcnt, el_GC, 70, 25, buffer);
sprintf(buffer, "Qes=%-5.1f", Qes);       drawString(wcnt, el_GC, 70, 55, buffer);
sprintf(buffer, "Tre=%-6.2f", Tre);       drawString(wcnt, el_GC, 70, 85, buffer);
sprintf(buffer, "Trs=%-6.2f", Trs);       drawString(wcnt, el_GC, 70, 115, buffer);
sprintf(buffer, "Wvsc=%-8.5f", Wvsc);     drawString(wcnt, el_GC, 230, 25, buffer);
sprintf(buffer, "Wb=%-8.5f", Wb);         drawString(wcnt, el_GC, 230, 55, buffer);
sprintf(buffer, "Wr=%-8.5f", Wr);         drawString(wcnt, el_GC, 230, 85, buffer);
sprintf(buffer, "kx=%-7.5f", kx);         drawString(wcnt, el_GC, 230, 115, buffer);
}
/*=========================================*/
void abrir_ventanas_esquema3(double tf)
{
wd31=abroVentana(5, 135, 280, 250, VENTANA_POP_UP, "wd31", 0, ventana_raiz, &el_GC);
wd32=abroVentana(290, 135, 280, 250, VENTANA_POP_UP, "wd32", 0, ventana_raiz, &el_GC);

wd33=abroVentana(5, 390, 280, 250, VENTANA_POP_UP, "wd33", 0, ventana_raiz, &el_GC);
wd34=abroVentana(290, 390, 280, 250, VENTANA_POP_UP, "wd34", 0, ventana_raiz, &el_GC);
wd35=abroVentana(575, 390, 280, 250, VENTANA_POP_UP, "wd35", 0, ventana_raiz, &el_GC);
wd36=abroVentana(860, 390, 280, 250, VENTANA_POP_UP, "wd36", 0, ventana_raiz, &el_GC);

wd37=abroVentana(5, 645, 280, 250, VENTANA_POP_UP, "wd37", 0, ventana_raiz, &el_GC);
wd38=abroVentana(290, 645, 280, 250, VENTANA_POP_UP, "wd38", 0, ventana_raiz, &el_GC);
wd39=abroVentana(575, 645, 280, 250, VENTANA_POP_UP, "wd39", 0, ventana_raiz, &el_GC);
wd310=abroVentana(860, 645, 280, 250, VENTANA_POP_UP, "wd310", 0, ventana_raiz, &el_GC);

dibujo_entorno(wd31, "Niveles (m)", "SkyBlue", 2, "nc", "ncd", "", "", "Black", "Yellow", "", "", 1., tf);
dibujo_entorno(wd32, "Presiones (kPa)", "SkyBlue", 2, "Pc", "Pcd", "", "","Black","Yellow", "", "",
30000., tf);
dibujo_entorno(wd33, "Temperaturas I (C)", "SkyBlue", 2, "Tc", "Tcd", "", "", "Black", "Yellow", "", "",
                                                    926.87, tf);
dibujo_entorno(wd34, "Entalpias (kJ/kg)", "SkyBlue", 3, "hves", "hvss", "hvst", "", "Black", "White",
                                                    "Yellow", "", 4000., tf);
dibujo_entorno(wd35, "Volumen especifico (m3/kg)", "SkyBlue", 3, "vves", "vvss", "vvst", "", "Black",
                                                    "White", "Yellow", "", 2., tf);
dibujo_entorno(wd36, "Temperaturas II (C)", "SkyBlue", 3, "Tvss", "Tvst", "Tlsb", "", "White", "Yel-
low",
                                                    "Black", "", 926.87, tf);
dibujo_entorno(wd37, "Flujos de Calor (kJ/s)", "MediumAquamarine", 2, "Qec", "Qes", "", "", "Black",
                                                    "Yellow", "", "", 100., tf);
dibujo_entorno(wd38, "Caudales (kg/s)", "MediumAquamarine", 4, "Wvst", "Wvsc", "Wb", "Wr",
"White",
                                                    "Black", "Red", "Yellow", 0.1, tf);
dibujo_entorno(wd39, "Temperat III (C)", "MediumAquamarine", 2, "Tre", "Trs", "", "", "White",
"Black",
                                                    "", "", 100., tf);
dibujo_entorno(wd310, "kx y Rendimiento (s.d.)", "GreenYellow", 2, "kx", "R", "", "", "Black", "Blue",
"", "",
                                                    1., tf);
}
/*=========================================*/
voidcierro_ventanas_esquema3(void)
{
XDestroyWindow(el_display, wd31);
XDestroyWindow(el_display, wd32);
XDestroyWindow(el_display, wd33);
XDestroyWindow(el_display, wd34);
XDestroyWindow(el_display, wd35);
```

```
XDestroyWindow(el_display, wd36);
XDestroyWindow(el_display, wd37);
XDestroyWindow(el_display, wd38);
XDestroyWindow(el_display, wd39);
XDestroyWindow(el_display, wd310);
XFlush(el_display);
}
```

eventos.c

```
/* eventos.c */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>

#include "windows.h"
/*=========================================*/
extern Display *el_display;
#define EV_MASK ( ButtonPressMask | KeyPressMask | ExposureMask | StructureNotifyMask )
/*=========================================*/
void inicia_eventos(Window theWindow)
    {
    XSelectInput(el_display, theWindow, EV_MASK);
    }
```

funcions.c

```
/* funcions.c */
/*=========================================*/
#include "clases.h"
/*=========================================*/
double intrplcn(double x1, double x2, double y1, double y2, double x0)
    {
    double res_intrplcn=y1+(x0-x1)*(y2-y1)/(x2-x1);
    return(res_intrplcn);
    }
/*-----------------------------------------------------------------*/
double tabla_1d(double x0, double y[], double x[], int size)
    {
    double y0; int i;

    for(i=0; i<=size-1; i++)
        if(x0==x[i])
            { y0=y[i];i =size;}
        else if(x0>x[i] && x0<x[i+1])
            { y0=y[i]+(x0-x[i])*(y[i+1]-y[i])/(x[i+1]-x[i]); i=size; }
    return(y0);
    }
/*-----------------------------------------------------------------*/
double tabla_2d_xy( double x[], int xsize, double y[], int ysize, double z[], double x0, double y0)
{
/*Importante:los elementos de los arrays x[], y[], z[n][], han de ser crecientes*/
int i, j, ci, cj, inx, iny;          /* modo de llamar a esta funcin:   */
double z0;                           /* double *arr;                    */
                                     /* arr=&z[0][0]                    */
for(i=0; i<xsize; i++)     /* z0=tabla_2d_xy(x, 9, y, 8, arr, x0, y0);*/
    {
    if(x0==x[i])
        { ci=i; inx=0; i=xsize; }
    else if(x[i]<x0 && x0<x[i+1])
        { ci=i; inx=1; i=xsize; }
    }
for(j=0; j<ysize; j++)
    {
    if(y0==y[j])
        { cj=j; iny=0; j=ysize; }
    else if(y[j]<y0 && y0<y[j+1])
        { cj=j; iny=1; j=ysize; }
    }
if(inx==0 && iny==0)
    { z0=z[ysize*ci+cj]; }
if(inx==1 && iny==0)
    { z0=intrplcn(x[ci], x[ci+1], z[ysize*ci+cj], z[ysize*(ci+1)+cj], x0); }
if(inx==0 && iny==1)
    { z0=intrplcn(y[cj], y[cj+1], z[ysize*ci+cj], z[ysize*ci+cj+1], y0); }
if(inx==1 && iny==1)
    { z0=intrplcn(y[cj], y[cj+1], intrplcn(x[ci], x[ci+1], z[ysize*ci+cj], z[ysize*(ci+1)+cj], x0),
          intrplcn(x[ci], x[ci+1], z[ysize*ci+cj+1], z[ysize*(ci+1)+cj+1], x0), y0); }
return(z0);
}
/*-----------------------------------------------------------------*/
double tabla_2d_xz( double x[], int xsize, double y[], int ysize, double z[], double x0, double z0)
/*Importante:los elementos delos arrays x[], y[], z[n][], han de ser crecientes*/
{
int i, j, ci;                        /* modo de llamar a esta funci<162n:   */
double y0, a, b;                     /* double *arr;                    */
                                     /* arr=&z[0][0]                    */
for(i=0; i<xsize; i++)     /* y0=tabla_2d_xz(x, 9, y, 8, arr, x0, z0);*/
    {
    if(x0==x[i])
```

```
        { ci=i; i=xsize; }
    else if(x[i]<x0 && x0<x[i+1])
        { ci=i; i=xsize; }
    }
for(j=0; j<ysize; j++)
    {
    a=intrplcn(x[ci], x[ci+1], z[ysize*ci+j],  z[ysize*(ci+1)+j],  x0);
    b=intrplcn(x[ci], x[ci+1], z[ysize*ci+j+1], z[ysize*(ci+1)+j+1], x0);
    if(z0==a)
        { y0=y[j]; }
    else if(a<z0 && z0<b)
        { y0=intrplcn(a, b, y[j], y[j+1], z0); }
    }
return(y0);
}
/*-------------------------------------------------------------------*/
double tabla_2d_xz_sh( double x[], int xsize, double y[], int ysize, double z[], double x0, double z0)
{
/*Importante:los elementos delos arrays x[], y[], z[n][], han de ser crecientes*/
/* Modo de usar esta funci<162n:                    */
/* double P[5]=..., T[5][7]=..., h[5][7]=...;       */
/* main() {                                         */
/* double P0, h0, T0, *arrT, *arrh;                 */
/* arrh=&h[0][0]; arrT=&T[0][0]; P0=...; h0=...;    */
/* T0=tabla_2d_xz_2(P, 5, arrT, 7, arrh, P0, h0); }*/
/* OJO: dimensionar yint[], zint[] como m<161nimo al */
/* valor de ysize.                                  */

int i, j, ci, inx, jinit, iN;
double y0, yint[19], zint[19];
for(i=0; i<=ysize; i++)
    { yint[i]=0.0; zint[i]=0.0; }
for(i=0;i<xsize;i++)
    {
    if(x0==x[i])
        { ci=i; inx=0; i=xsize; }
    else if(x[i]<x0 && x0<x[i+1])
        { ci=i; inx=1; i=xsize; }
    }
if(inx==0)
    {
    for(j=0; j<ysize; j++)
        {
        if(z0==z[ysize*ci+j])
            { y0=y[ysize*ci+j]; j=ysize; }
        else if(z[ysize*ci+j]<z0 && z0<z[ysize*ci+j+1])
            {
            y0=intrplcn(z[ysize*ci+j], z[ysize*ci+j+1], y[ysize*ci+j], y[ysize*ci+j+1], z0);
            j=ysize;
            }
        }
    }
else {
    yint[0]=intrplcn(x[ci], x[ci+1], y[ysize*ci], y[ysize*(ci+1)], x0);
    zint[0]=intrplcn(x[ci], x[ci+1], z[ysize*ci], z[ysize*(ci+1)], x0);
    jinit=1;
    for(i=1; i<ysize; i++)
        {
        if(y[ysize*(ci+1)+i]!=0)
            {
            for(j=jinit; j<ysize; j++)
                {
                if(y[ysize*ci+j]==y[ysize*(ci+1)+i])
                    {
                    yint[i]=y[ysize*ci+j];
                    zint[i]=intrplcn(x[ci], x[ci+1], z[ysize*ci+j], z[ysize*(ci+1)+i], x0);
                    jinit=j+1;
                    j=ysize;
                    }
                }
            }
        else    {
            if(y[ysize*ci+jinit]==0)
                { iN=i-1; i=ysize; }
            else {
                iN=i;
                yint[i]=intrplcn(x[ci], x[ci+1], y[ysize*ci+jinit], y[ysize*(ci+1)+i-1], x0);
                zint[i]=intrplcn(x[ci], x[ci+1], z[ysize*ci+jinit], z[ysize*(ci+1)+i-1], x0);
                i=ysize;
                }
            }
        }
    for(i=0; i<=iN; i++)
        {
        if(z0==zint[i])
            { y0=yint[i]; i=iN+1; }
        else if(zint[i]<z0 && z0<zint[i+1])
            {
```

```
    yint[0]=intrplcn(x[ci], x[ci+1], y[ysize*ci], y[ysize*(ci+1)], x0);
    zint[0]=intrplcn(x[ci], x[ci+1], z[ysize*ci], z[ysize*(ci+1)], x0);
    jinit=1;
    for(i=1; i<ysize; i++)
        {
        if(y[ysize*(ci+1)+i]!=0)
            {
            for(j=jinit;j<ysize;j++)
                {
                if(y[ysize*ci+j]==y[ysize*(ci+1)+i])
                    {
                    yint[i]=y[ysize*ci+j];
                    zint[i]=intrplcn(x[ci], x[ci+1], z[ysize*ci+j], z[ysize*(ci+1)+i], x0);
                    jinit=j+1;
                    j=ysize;
                    }
                }
            }
        else    {
            if(y[ysize*ci+jinit]==0)
                { iN=i-1; i=ysize; }
            else {
                iN=i;
                yint[i]=intrplcn(x[ci], x[ci+1], y[ysize*ci+jinit], y[ysize*(ci+1)+i-1], x0);
                zint[i]=intrplcn(x[ci], x[ci+1], z[ysize*ci+jinit], z[ysize*(ci+1)+i-1], x0);
                i=ysize;
                }
            }
        }
    for(i=0; i<=iN; i++)
        {
        if(y0==yint[i])
            { z0=zint[i]; i=iN+1; }
        else if(yint[i]<y0 && y0<yint[i+1])
            { z0=intrplcn(yint[i], yint[i+1], zint[i], zint[i+1], y0); i=iN+1; }
        }
    }
return(z0);
}
/*-----------------------------------------------------------------*/
double tabla_2d_xy_liq( double x[], int xsize, double y[], int ysize, double z[], double x0, double y0)
{
/* Modo de usar esta funci<162n:                    */
/* double P[4]=..., T[4][5]=..., h[4][5]=...;       */
```



```
            y0=intrplcn(zint[i], zint[i+1], yint[i], yint[i+1], z0);
            i=iN+1;
            }
        }
    }
return(y0);
}
/*-----------------------------------------------------------------*/
double tabla_2d_xy_sh( double x[], int xsize, double y[], int ysize, double z[], double x0, double y0)
{
/*Importante:los elementos delos arrays x[], y[], z[n][], han de ser crecientes*/
/* Modo de usar esta funci<162n:                                  */
/* double P[5]=..., T[5][7]=..., v[5][7]=...;                     */
/* main() {                                                       */
/* double P0, T0, v0, *arrT, *arrv;                               */
/* arrv=&v[0][0]; arrT=&T[0][0]; P0=...; T0=...;                  */
/* v0=tabla_2d_xy_2(P, 5, arrT, 7, arrv, P0, T0); }               */
/* OJO: dimensionar yint[], zint[] como mnimo al valor de ysize.*/

int i, j, ci, cj, inx, iny, jinit, iN;
double z0, yint[19], zint[19];

for(i=0; i<=ysize; i++)
    { yint[i]=0.0; zint[i]=0.0; }
for(i=0;i<xsize;i++)
    {
    if(x0==x[i])
        { ci=i; inx=0; i=xsize; }
    else if(x[i]<x0 && x0<x[i+1])
        { ci=i; inx=1; i=xsize; }
    }
for(j=0;j<ysize;j++)
    {
    if(y0==y[ysize*ci+j])
        { cj=j; iny=0; j=ysize; }
    else if(y[ysize*ci+j]<y0 && y0<y[ysize*ci+j+1])
        { cj=j; iny=1; j=ysize; }
    }
if(inx==0 && iny==0)
    z0=z[ysize*ci+cj];
if(inx==0 && iny==1)
    z0=intrplcn(y[ysize*ci+cj], y[ysize*ci+cj+1], z[ysize*ci+cj], z[ysize*ci+cj+1], y0);
if(inx==1)
    {
```

```
/* OJO: dimensionar yint[], zint[] como mnimo al valor de ysize. */
double y0, yint[22], zint[22];
int i, j, ci, inx, jN, jsN;

for(i=0; i<=ysize; i++)
    { yint[i]=0.0; zint[i]=0.0; }
for(i=0; i<xsize; i++)
    {
    if(x0==x[i])
        { ci=i; inx=0; i=xsize; }
    else if(x[i]<x0 && x0<x[i+1])
        { ci=i; inx=1; i=xsize; }
    }
for(j=0; j<ysize; j++)
    {
    if(y[ysize*ci+j]==0.0)
        { jN=j-1; j=ysize; }
    else if(j==ysize-1)
        { jN=j;  j=ysize; }
    }
for(j=0; j<ysize; j++)
    {
    if(y[ysize*(ci+1)+j]==0.0)
        { jsN=j-1; j=ysize; }
    else if(j==ysize-1)
        { jsN=j;  j=ysize; }
    }
if(inx==0)
    {
    for(j=0; j<=jN; j++)
        {
        if(z0==z[ysize*ci+j])
            { y0=y[ysize*ci+j]; j=jN+1; }
        else if(z[ysize*ci+j]<z0 && z0<z[ysize*ci+j+1])
            {
            y0=intrplcn(z[ysize*ci+j], z[ysize*ci+j+1], y[ysize*ci+j], y[ysize*ci+j+1], z0);
            j=jN+1;
            }
        }
    }
else {
    if(jN==jsN)
        {
        for(j=0; j<=jsN; j++)
            {
            yint[j]=intrplcn(x[ci], x[ci+1], y[ysize*ci+j], y[ysize*(ci+1)+j], x0);
            zint[j]=intrplcn(x[ci], x[ci+1], z[ysize*ci+j], z[ysize*(ci+1)+j], x0);
            }
        }
    else {
        for(j=0; j<=jsN; j++)
            {
            if(j<=jN)
                {
                yint[j]=intrplcn(x[ci], x[ci+1], y[ysize*ci+j], y[ysize*(ci+1)+j], x0);
                zint[j]=intrplcn(x[ci], x[ci+1], z[ysize*ci+j], z[ysize*(ci+1)+j], x0);
                }
            else
                {
                yint[j]=intrplcn(x[ci], x[ci+1], y[ysize*ci+jN], y[ysize*(ci+1)+j], x0);
                zint[j]=intrplcn(x[ci], x[ci+1], z[ysize*ci+jN], z[ysize*(ci+1)+j], x0);
                }
            }
        }
    for(j=0; j<=jsN; j++)
        {
        if(z0==zint[j])
            { y0=yint[j]; j=jsN+1; }
        else if(zint[j]<z0 && z0<zint[j+1])
            { y0=intrplcn(zint[j], zint[j+1], yint[j], yint[j+1], z0); j=jsN+1; }
        }
    }
return(y0);
}
```

initx.c

```
/* initx.c */          /* CODIGO DE INICIALIZACION PARA CONECTAR CON EL SERVIDOR X */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/cursorfont.h>
#include <stdio.h>
```

```
#include <stdlib.h>

#include "windows.h"
/*=========================================*/
Display     *el_display;        /* Cuál display */
int         el_screen;          /* Cuál screen sobre el display */
int         profundidad;        /* Número de planos de color */
unsigned long   pixel_negro;    /* color "Black" del Sistema */
unsigned long   pixel_blanco;   /* color "White" del Sistema */
Colormap    mapa_color;         /* mapa de color por defecto del Sistema  */
Cursor      el_cursor;          /* cursor del programa de aplicación */
Window      ventana_raiz;
/*=========================================*/
                                /* Establece la conexi<162n con el X Server*/
void initX(char *nombre_display)        /* y almacena informacin sobre el entorno*/
    {
    el_display = XOpenDisplay(nombre_display); /* conecta con X Server */
    if(el_display == NULL)              /* comprueba la conexi<162n */
        {
        fprintf (stderr, "ERROR:Cannot establish a connection to the X Server  %s\n",
                                        XDisplayName(nombre_display));
        exit( 1 );
        }
                    /* comprueba la screen  y el  n<163mero de planos de color */
                    /* por defecto. Si theDepht=1 , el sistema es monocromo  */
el_screen = DefaultScreen ( el_display );
profundidad  = DefaultDepth( el_display,  el_screen );
pixel_negro = BlackPixel (el_display,  el_screen);
pixel_blanco = WhitePixel (el_display,  el_screen);
mapa_color   = DefaultColormap( el_display,  el_screen);

                    /* crea un cursor para todas las ventanas del programa */
el_cursor = XCreateFontCursor(el_display, XC_cross);
ventana_raiz   = RootWindow( el_display,  el_screen);
}
/*=========================================*/
void info_X(void)                   /* Imprime informacin a la terminal sobre la */
    {                               /* display y la screen X Window actuales */
    printf("%s version %d of the Windows System , X%d R%d\n",
    ServerVendor( el_display ),
    VendorRelease( el_display ),
    ProtocolVersion( el_display ),
    ProtocolRevision( el_display ) );
    if(profundidad == 1)
        printf("Color plane depth.....%d(monochrome)\n", profundidad);
    else
        printf("color plane depth.....%d\n", profundidad);
    printf("Display width.....%d\n", DisplayWidth(el_display,el_screen));
    printf("DisplayHeight.....%d\n", DisplayHeight(el_display,  el_screen));
    /* printf("the display %s\n", XDisplayName(el_display) ); */
}
```

main.c

```
/** main.c **/
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
#include <string.h>

#include "windows.h"
/*=========================================*/
extern Display      *el_display;
GC                  el_GC;
extern Window       ventana_raiz;

Window      w, w0, w1, w2, w3, w4;
Window      wmm, ws, wm, wc, we;
Window      wp, wpc, wpk, wp00, wp01, wp10, wp11, wp20, wp21, wp30, wp31, wp40, wp41;
Window          wp50, wp51, wp60, wp61, wp70, wp71;
Window          wp02, wp03, wp12, wp13, wp22, wp23, wp32, wp33, wp42, wp43;
Window          wp52, wp53, wp62, wp63;
Window      wg, wgsg, wgng, wgc, wge, wgg, wgg0, wgg1, wgg2, wgg3, wggb, wget, wgeb;
Window      wcn, wcn1, wcn2, wcn3, wcn4;
Window      wcnt, wcnt00, wcnt01, wcnt10, wcnt11, wcnt20, wcnt21, wcnt30, wcnt31;
Window      wcnt40, wcnt41, wcnt50, wcnt51, wcnt60, wcnt61, wcnt70, wcnt71;
Window      wd11, wd12, wd13, wd14, wd15, wd16;
Window      wd21, wd22, wd23, wd24, wd25, wd26, wd27, wd28;
Window      wd31, wd32, wd33, wd34, wd35, wd36, wd37, wd38, wd39, wd310;
Window      wmt, wmtk, wmtc, wmta, wmt00, wmt01, wmtat, wmtab;
Window      wa, wa0, wa1, wa2, wa3, waa, wbaa;
Window      wam, wbam;
```

```
/*=========================================*/
main()
    {
    Scheduler juan;
    juan.start();
    }
/*=========================================*/
void Scheduler::start(void)
{
initX(nombre_display);
//info_X();
init_colores_defecto();

o1=0; o2=0; o3=0; o4=0; o5=0; o6=0;

abro_menu_inicial();
do  {
    o1=menu_inicial();
    if(o1==4)       ayuda();
    if(o1==1 || o1==2 || o1==3)
        {
        abro_menu_modo();
        do  {
            o2=menu_modo();
            if(o2==3)       ayuda_modo();
            if(o2==1 || o2==2)
                {
                switch(o2)
                    {
                    case 1: abro_menu_parametros();
                            do  {
                                o3=menu_parametros();
                                if(o3==1)
                                    {
                                    abro_menu_grabacion();
                                    do  {
                                        o5=menu_grabacion();
                                        if(o5==3)       ayuda_grabacion();
                                        if(o5==1)       set_grabacion();
                                        if(o5==1 || o5==2)
                                            {
                                            abro_control();
                                            do  {
                                                control();
                                                }
                                                while(o6!=0);
                                            cierro_control();
                                            }
                                        }
                                        while(o5!=0);
                                    cierro_menu_grabacion();
                                    }
                                }
                                while(o3!=0);
                            cierro_menu_parametros();
                            break;
                    case 2: abro_menu_mostrar();
                            do  {
                                o4=menu_mostrar();
                                if(o4==2)
                                        ayuda_mostrar();
                                if(o4==3 || o4==4)
                                        set_fichero_mostrar();
                                if(o4==1)
                                        muestra_simulacion();
                                }
                                while(o4!=0);
                            cierro_menu_mostrar();
                            break;
                    default: break;
                    }
                }
            }
            while(o2!=0);
        cierro_menu_modo();
        }
    }
    while(o1!=0);
XFlush(el_display);
quitX();
}
```

menus.c

```
/* menus.c */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/stat.h>
#include<../../../home/gnu/gcc/libg++-1.39.0/g++-include/builtin.h>

#include "windows.h"
/*=========================================*/
extern Window  w, w0, w1, w2, w3, w4;
extern Window  wmm, ws, wm, wc, we;
extern Window  wp, wpc, wpk, wp00, wp01, wp10, wp11, wp20, wp21, wp30, wp31, wp40, wp41;
extern Window  wp50, wp51, wp60, wp61, wp70, wp71;
extern Window  wp02, wp03, wp12, wp13, wp22, wp23, wp32, wp33, wp42, wp43;
extern Window  wp52, wp53, wp62, wp63;
extern Window  wg, wgsg, wgng, wgc, wge, wgg, wgg0, wgg1, wgg2, wgg3, wggb;
extern Window  wcn, wcn1, wcn2, wcn3, wcn4, wd1;
extern Window  wcnt;
extern Window  wmt, wmtk, wmtc, wmta, wmt00, wmt01;
extern Window  wd11, wd12, wd13;

extern Window  ventana_raiz;
extern GC      el_GC;
extern Display *el_display;
/*=========================================*/
voidScheduler::abro_menu_inicial(void)
{
w=abroVentana(50, 50, 550, 400, VENTANA_POP_UP, "CICLOS", ESTADO_NORMAL,
                                        ventana_raiz, &el_GC);
inicia_eventos(w);
w0=abroVentana(5, 5, 90, 40, VENTANA_NORMAL, "w0", ESTADO_NORMAL, w, &el_GC);
inicia_eventos(w0);
w1=abroVentana(5, 80, 90, 40, VENTANA_NORMAL, "w1", ESTADO_NORMAL, w, &el_GC);
inicia_eventos(w1);
w2=abroVentana(5, 130, 90, 40, VENTANA_NORMAL, "w2", ESTADO_NORMAL, w, &el_GC);
inicia_eventos(w2);
w3=abroVentana(5, 180, 90, 40, VENTANA_NORMAL, "w3", ESTADO_NORMAL, w, &el_GC);
inicia_eventos(w3);
w4=abroVentana(5, 255, 90, 40, VENTANA_NORMAL, "w4", ESTADO_NORMAL, w, &el_GC);
inicia_eventos(w4);

cursores();       XFlush(el_display);
expose_menu_inicial();
}
/*=========================================*/
intScheduler::menu_inicial(void)
{
XEvent theEvent;
int       op;

XNextEvent(el_display,&theEvent);
switch(theEvent.type)
    {
    case ButtonPress:    op=raton_menu_inicial(&theEvent.xbutton)break;
    case KeyPress:       op=teclado_menu_inicial(&theEvent.xkey)break;
    default:             op=10;                              break;
    }
XFlush(el_display);
return(op);
}
/*=========================================*/
void Scheduler::abro_menu_modo(void)
{
wmm=abroVentana(180, 90, 250, 300, VENTANA_POP_UP, "Menu de modo", ESTADO_NORMAL,
                                        ventana_raiz, &el_GC);
inicia_eventos(wmm);
ws=abroVentana(10, 50, 90, 40, VENTANA_NORMAL, "ws", ESTADO_NORMAL, wmm, &el_GC);
inicia_eventos(ws);
wm=abroVentana(10, 100, 90, 40, VENTANA_NORMAL, "wm", ESTADO_NORML, wmm, &el_GC);
inicia_eventos(wm);
wc=abroVentana(150, 50, 90, 40, VENTANA_NORMAL, "wc", ESTADO_NORMAL, wmm, &el_GC);
inicia_eventos(wc);
we=abroVentana(150, 100, 90, 40, VENTANA_NORMAL, "we", ESTADO_NORML, wmm, &el_GC);
inicia_eventos(we);

expose_menu_modo();
}
/*=========================================*/
int Scheduler::menu_modo(void)
{
XEvent theEvent;
int       op;
```

```
    XNextEvent(el_display, &theEvent);
    switch(theEvent.type)
        {
        case ButtonPress:       op=raton_menu_modo(&theEvent.xbutton);break;
        case KeyPress:          op=teclado_menu_modo(&theEvent.xkey);break;
        default:                op=10;                              break;
        }

    XFlush(el_display);
    return(op);
    }
    /*=========================================*/
    void Scheduler::cierro_menu_modo(void)
    {
    XDestroySubwindows(el_display, wmm);
    XDestroyWindow(el_display, wmm);

    XFlush(el_display);
    }
    /*=========================================*/
    void Scheduler::abro_menu_parametros(void)
    {
    wp=abroVentana(225, 125, 450, 350, VENTANA_POP_UP, "Menu de parametros",
                                ESTADO_NORMAL, ventana_raiz, &el_GC);
    inicia_eventos(wp);

    switch(o1)
        {
    case 3:
    wp40=abroVentana(10, 170, 20, 20, VENTANA_NORMAL, "w40", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp40);
    wp41=abroVentana(35, 170, 20, 20, VENTANA_NORMAL, "w41", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp41);
    wp50=abroVentana(10, 200, 20, 20, VENTANA_NORMAL, "w50", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp50);
    wp51=abroVentana(35, 200, 20, 20, VENTANA_NORMAL, "w51", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp51);
    wp60=abroVentana(10, 230, 20, 20, VENTANA_NORMAL, "w60", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp60);
    wp61=abroVentana(35, 230, 20, 20, VENTANA_NORMAL, "w61", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp61);
    wp70=abroVentana(10, 260, 20, 20, VENTANA_NORMAL, "w70", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp70);
    wp71=abroVentana(35, 260, 20, 20, VENTANA_NORMAL, "w71", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp71);
    wp02=abroVentana(230, 50, 20, 20, VENTANA_NORMAL, "w02", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp02);
    wp03=abroVentana(255, 50, 20, 20, VENTANA_NORMAL, "w03", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp03);
    wp12=abroVentana(230, 80, 20, 20, VENTANA_NORMAL, "w12", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp12);
    wp13=abroVentana(255, 80, 20, 20, VENTANA_NORMAL, "w13", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp13);
    wp22=abroVentana(230, 110, 20, 20, VENTANA_NORMAL, "w22", ESTADO_NORML, wp,
&el_GC);
        inicia_eventos(wp22);
    wp23=abroVentana(255, 110, 20, 20, VENTANA_NORMAL, "w23", ESTADO_NORML, wp,
&el_GC);
        inicia_eventos(wp23);
    wp32=abroVentana(230, 140, 20, 20, VENTANA_NORMAL, "w32", ESTADO_NORML, wp,
&el_GC);
        inicia_eventos(wp32);
    wp33=abroVentana(255, 140, 20, 20, VENTANA_NORMAL, "w33", ESTADO_NORML, wp,
&el_GC);
        inicia_eventos(wp33);
    wp42=abroVentana(230, 170, 20, 20, VENTANA_NORMAL, "w42", ESTADO_NORML, wp,
&el_GC);
        inicia_eventos(wp42);
    wp43=abroVentana(255, 170, 20, 20, VENTANA_NORMAL, "w43", ESTADO_NORML, wp,
&el_GC);
        inicia_eventos(wp43);
    wp52=abroVentana(230, 200, 20, 20, VENTANA_NORMAL, "w52", ESTADO_NORML, wp,
&el_GC);
        inicia_eventos(wp52);
    wp53=abroVentana(255, 200, 20, 20, VENTANA_NORMAL, "w53", ESTADO_NORML, wp,
&el_GC);
```

```
        inicia_eventos(wp53);
    wp62=abroVentana(230, 230, 20, 20, VENTANA_NORMAL, "w62", ESTADO_NORML, wp,
&el_GC);
        inicia_eventos(wp62);
    wp63=abroVentana(255, 230, 20, 20, VENTANA_NORMAL, "w63", ESTADO_NORML, wp,
&el_GC);
        inicia_eventos(wp63);
    case 2:
    case 1:
    wp00=abroVentana(10, 50, 20, 20, VENTANA_NORMAL, "w00", ESTADO_NORMAL, wp, &el_GC);
        inicia_eventos(wp00);
    wp01=abroVentana(35, 50, 20, 20, VENTANA_NORMAL, "w01", ESTADO_NORMAL, wp, &el_GC);
        inicia_eventos(wp01);
    wp10=abroVentana(10, 80, 20, 20, VENTANA_NORMAL, "w10", ESTADO_NORMAL, wp, &el_GC);
        inicia_eventos(wp10);
    wp11=abroVentana(35, 80, 20, 20, VENTANA_NORMAL, "w11", ESTADO_NORMAL, wp, &el_GC);
        inicia_eventos(wp11);
    wp20=abroVentana(10, 110, 20, 20, VENTANA_NORMAL, "w20", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp20);
    wp21=abroVentana(35, 110, 20, 20, VENTANA_NORMAL, "w21", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp21);
    wp30=abroVentana(10, 140, 20, 20, VENTANA_NORMAL, "w30", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp30);
    wp31=abroVentana(35, 140, 20, 20, VENTANA_NORMAL, "w31", ESTADO_NORMAL, wp,
&el_GC);
        inicia_eventos(wp31);
        break;
    default: break;
        }

    wpk=abroVentana(10, 300, 90, 40, VENTANA_NORMAL, "wpk", ESTADO_NORMAL, wp, &el_GC);
    inicia_eventos(wpk);
    wpc=abroVentana(110, 300, 90, 40, VENTANA_NORMAL, "wpc", ESTADO_NORMAL, wp,
&el_GC);
    inicia_eventos(wpc);

    if(o5==0 && o3==1)
        { }
    else
        {
        switch(o1)
        {
        case 1:  h=0.5; t0=0.; tf=300.; P_caldera=4000.;
                 break;
        case 2:  h=0.5; t0=0.; tf=300.; P_caldera=4000.;
                 break;
        case 3:  h=0.5;t0=0.; tf=300.; P_caldera=4000.;
                 P_condensador=10.; W_refrigerante=0.005;
                 T_refrig_in=20.; T_refrig_out=25.;
                 m_refrigerante=1.; W_bomba=0.005;
                 W_valvula1=0.005; kx_turbina=0.1;
                 h_supercal=2850.; m_supercal=1.; k_supercal=10.;
                 break;
        default: break;
        }
    }
expose_menu_parametros(o1);
}
/*=============================================*/
int Scheduler::menu_parametros(void)
{
XEvent theEvent;
int    op;

imprimo_parametros();

XNextEvent(el_display, &theEvent);

switch(theEvent.type)
    {
    case ButtonPress:   op=raton_menu_parametros(&theEvent.xbutton);break;
    case KeyPress:      op=teclado_menu_parametros(&theEvent.xkey);break;
    default:            op=10;                                  break;
    }
XFlush(el_display);
return(op);
}
/*=============================================*/
void Scheduler::cierro_menu_parametros(void)
{
XDestroySubwindows(el_display, wp);
XDestroyWindow(el_display, wp);
XFlush(el_display);
}
/*=============================================*/
```

```
void Scheduler::abro_menu_mostrar(void)
{
wmt=abroVentana(600, 50, 220, 330, VENTANA_POP_UP, "Menu de mostrar",
                                    ESTADO_NORMAL, ventana_raiz, &el_GC);
inicia_eventos(wmt);
wmtk=abroVentana(10, 50, 90, 40, VENTANA_NORMAL, "wmtk", ESTADO_NORMAL, wmt,
&el_GC);
inicia_eventos(wmtk);
wmtc=abroVentana(120, 50, 90, 40, VENTANA_NORMAL, "wmtc", ESTADO_NORMAL, wmt,
&el_GC);
inicia_eventos(wmtc);
wmta=abroVentana(65, 100, 90, 40, VENTANA_NORMAL, "wmta", ESTADO_NORMAL, wmt,
&el_GC);
inicia_eventos(wmta);

wmt00=abroVentana(20, 260, 20, 20, VENTANA_NORMAL, "wmt00", ESTADO_NORMAL, wmt,
&el_GC);
inicia_eventos(wmt00);
wmt01=abroVentana(180, 260, 20, 20, VENTANA_NORMAL, "wmt01", ESTADO_NORMAL,wmt,
&el_GC);
inicia_eventos(wmt01);

file_mostrar=1;
expose_menu_mostrar();
set_fichero_mostrar();
}
/*=========================================*/
int Scheduler::menu_mostrar(void)
{
XEvent theEvent;
int    op;

XNextEvent(el_display,&theEvent);
switch(theEvent.type)
    {
    case ButtonPress:   op=raton_menu_mostrar(&theEvent.xbutton);break;
    case KeyPress:      op=teclado_menu_mostrar(&theEvent.xkey);break;
    default:            op=10;                              break;
    }
XFlush(el_display);
return(op);
}
/*=========================================*/
void Scheduler::cierro_menu_mostrar(void)
{
XDestroySubwindows(el_display, wmt);
XDestroyWindow(el_display, wmt);

XFlush(el_display);
}
/*=========================================*/
void Scheduler::set_fichero_mostrar(void)
{
char *cx, *cy;          /* simx_y.dat */

if(o4==4)       file_mostrar=file_mostrar+1;
if(o4==3)       { if(file_mostrar>1)  file_mostrar=file_mostrar-1; }

XClearArea(el_display, wmt, 55, 255, 110, 30, 0);
XClearArea(el_display, wmt, 55, 305, 110, 10, 0);

cx=itoa(o1, 10, 0);
cy=itoa(file_mostrar, 10, 0);
strcpy(filename, "sim");
strcat(filename, cx);
strcat(filename, "_");
strcat(filename, cy);
strcat(filename, ".dat");

drawString(wmt, el_GC, 70, 275, filename);
if((fp=fopen(filename, "r"))==NULL)      drawString(wmt, el_GC, 65, 315, "NO DISPONIBLE");
else                                    drawString(wmt, el_GC, 65, 315, "SI DISPONIBLE");
fclose(fp);
}
/*=========================================*/
void Scheduler::abro_menu_grabacion(void)
{
wg=abroVentana(460, 160, 400, 300, VENTANA_POP_UP, "Menu de grabacion", ESTADO_NORML,
                                    ventana_raiz, &el_GC);
inicia_eventos(wg);
wgsg=abroVentana(10, 50, 90, 40, VENTANA_NORMAL, "wgsg", ESTADO_NORML, wg,
&el_GC);
inicia_eventos(wgsg);
wgng=abroVentana(10, 100, 90, 40, VENTANA_NORMAL, "wgng", ESTADO_NORML, wg,
&el_GC);
inicia_eventos(wgng);
wgc=abroVentana(10, 200, 90, 40, VENTANA_NORMAL, "wgc", ESTADO_NORML, wg, &el_GC);
inicia_eventos(wgc);
```

```
    wge=abroVentana(10, 250, 90, 40, VENTANA_NORMAL, "wge", ESTADO_NORMAL, wg, &el_GC);
    inicia_eventos(wge);

    expose_menu_grabacion();
    }
    /*=======================================*/
    intScheduler::menu_grabacion(void)
    {
    XEvent theEvent;
    int       op;

    XNextEvent(el_display,&theEvent);
    switch(theEvent.type)
        {
        case ButtonPress:    op=raton_menu_grabacion(&theEvent.xbutton)break;
        case KeyPress:       op=teclado_menu_grabacion(&theEvent.xkey)break;
        default:             op=10;                                  break;
        }
    XFlush(el_display);
    return(op);
    }
    /*=======================================*/
    voidScheduler::cierro_menu_grabacion(void)
    {
    XDestroySubwindows(el_display, wg);
    XDestroyWindow(el_display, wg);
    XFlush(el_display);
    }
    /*=======================================*/
    voidScheduler::set_grabacion(void)
    {
    XEvent Event;
    int       op;
    char      buffer[50];

    wgg=abroVentana(110, 50, 280, 110, VENTANA_NORMAL, "wgg", ESTADO_NORML, wg,
&el_GC);
    inicia_eventos(wgg);

    wgg0=abroVentana(10, 10, 20, 20, VENTANA_NORMAL, "wgg0", ESTADO_NORML, wgg,
&el_GC);
    inicia_eventos(wgg0);
    wgg1=abroVentana(35, 10, 20, 20, VENTANA_NORMAL, "wgg1", ESTADO_NORML, wgg,
&el_GC);
    inicia_eventos(wgg1);
    wgg2=abroVentana(10, 80, 20, 20, VENTANA_NORMAL, "wgg2", ESTADO_NORML, wgg,
&el_GC);
    inicia_eventos(wgg2);
    wgg3=abroVentana(35, 80, 20, 20, VENTANA_NORMAL, "wgg3", ESTADO_NORML, wgg,
&el_GC);
    inicia_eventos(wgg3);

    wggb=abroVentana(230, 60, 40, 40, VENTANA_NORMAL, "wggb", ESTADO_NORM, wgg,
&el_GC);
    inicia_eventos(wggb);

    boton2(wggb, "O", "K", "Red");
    boton3(wgg0, "LightBlue", "Yellow");  boton4(wgg1, "LightBlue", "Yellow");
    boton3(wgg2, "LightBlue", "Yellow");  boton4(wgg3, "LightBlue", "Yellow");

    hit=10.;
    num_fich=1;
    num_iter=(tf-t0)/h;
    N_grupos_dat=num_iter/hit;

    do  {
        XClearArea(el_display, wgg, 60, 10, 165, 90, 0);
        sprintf(buffer, "hit = %-5.1f", hit); drawString(wgg, el_GC, 70, 25, buffer);
        sprintf(buffer, "num_fich = %d", num_fich);
        drawString(wgg, el_GC, 70, 95, buffer);
        sprintf(buffer, "num_iter=%-9.0f", num_iter);
        drawString(wgg, el_GC, 70, 50, buffer);
        sprintf(buffer, "N_grupos=%-6.0f", N_grupos_dat);
        drawString(wgg, el_GC, 70, 65, buffer);

        XNextEvent(el_display, &Event);
        switch(Event.type)
            {
            case ButtonPress:       op=raton_set_grabacion(&Event.xbutton)break;
            case KeyPress:          op=teclado_botonOK(&Event.xkey)break;
            default:                op=1;                                break;
            }
        num_iter=(tf-t0)/h;
        N_grupos_dat=num_iter/hit;
        }
        while(op!=0);

    XDestroySubwindows(el_display, wgg);
```

```
#include <X11/Xutil.h>
#include <stdio.h>
#include <stdlib.h>

#include "windows.h"
/*=========================================*/
extern Display   *el_display;
extern Window  wd11, wd12, wd13, wd14, wd15, wd16;
extern Window  wd21, wd22, wd23, wd24, wd25, wd26, wd27, wd28;
extern Window  wd31, wd32, wd33, wd34, wd35, wd36, wd37, wd38, wd39, wd310;
/*=========================================*/
void Scheduler::muestra_simulacion(void)
{
XEvent Event;

if((fp=fopen(filename,"r"))==NULL)      { o4=0; } /* Abro el fichero a mostrar */
if(o4!=0)
    {
    esquema_mostrar(); /* Muestro los gr<160ficos */
    do  {
        XNextEvent(el_display,&Event);
        switch(Event.type)
            {
            case ButtonPress:       o4=raton_menu_mostrar(&Event.xbutton);break;
            default:                o4=10;                  break;
            }
        }
        while(o4!=0);
    switch(o1)      /* Cierro los graficos */
        {
        case 1:  cierro_ventanas_esquema1();break;
        case 2:  cierro_ventanas_esquema2();break;
        case 3:  cierro_ventanas_esquema3();break;
        default: break;
        }
    }
o4=10;
fclose(fp);     /* Cierro el fichero a motrar */
XFlush(el_display);
}
/*=========================================*/
void Scheduler::esquema_mostrar(void)
{
int  i;
char        texto[500];
double  dato;
double  Lc, Lcd, Rc, Rcd, mvs, ksup, mr, R;
double  t, nc, ncd, Pc, Pcd, Tc, Tcd, dlc, hlc, slc, mlc, dvc, hvc, svc, mvc;
double  Wlec, Wvsc, Wb, Wvst, Wr, kx;
double  Qec, Qes, hlec, hvst, vvst, Tvst, Tlsb, Trs, Tre;
double  Pves, Pvss, Tves, Tvss, hves, hvss, vves, vvss, sves, svss;
double  tf, N_grupos, cont;
double  u1d1, v1d1, u1d2, v1d2, u1d3, v1d3, u1d4, v1d4, u1d5, v1d5, u1d6, v1d6, u1d7, v1d7;
double  u1d8, v1d8, u1d9, v1d9, u1d10, v1d10;

fscanf(fp, "%s", texto);        /*leo "\nDatos_iniciales:....\n"*/
fscanf(fp, "%lf ", &dato);      /*leo t0*/
fscanf(fp, "%lf ", &tf);        /*leo tf*/
fscanf(fp, "%lf ", &dato);      /*leo h*/
fscanf(fp, "%lf ", &dato);      /*leo hit*/
fscanf(fp, "%lf ", &dato);      /*leo num_iter*/
fscanf(fp, "%lf ", &N_grupos);  /*leo N_grupos_dat*/

switch(o1)
    {
    case 1:abrir_ventanas_esquema1(tf);
        fscanf(fp, "%s", texto); /*leo"\nDatos_caract.:..\n"*/
        fscanf(fp, "%lf ", &Lc);
        fscanf(fp, "%s", texto); /*leo"\nDatos:t, nc, Pc, Tc, ...\n"*/

        u1d1 = u1d2 = u1d3 = u1d4 = u1d5 = u1d6 = tf;
        v1d1=Lc; v1d2=22090.; v1d3=374.5;
        v1d4=200.; v1d5=2000.; v1d6=0.1;

        for(cont=0.; cont<N_grupos; cont=cont+1.)
            {
            fscanf(fp, "%lf ", &t);         fscanf(fp, "%lf ", &nc);
            fscanf(fp, "%lf ", &Pc);        fscanf(fp, "%lf ", &Tc);
            fscanf(fp, "%lf ", &dlc);       fscanf(fp, "%lf ", &hlc);
            fscanf(fp, "%lf ", &slc);       fscanf(fp, "%lf ", &mlc);
            fscanf(fp, "%lf ", &dvc);       fscanf(fp, "%lf ", &hvc);
            fscanf(fp, "%lf ", &svc);       fscanf(fp, "%lf ", &mvc);
            fscanf(fp, "%lf ", &Wlec);          fscanf(fp, "%lf ", &Wvsc);
            fscanf(fp, "%lf ", &Qec);       fscanf(fp, "%lf ", &hlec);

            cambio_coordenadas(t, nc, u1d1, v1d1);   dibujo_punto(wd11, x, y, "Black");
            cambio_coordenadas(t, Pc, u1d2, v1d2);   dibujo_punto(wd12, x, y, "Yellow");
            cambio_coordenadas(t, Tc, u1d3, v1d3);   dibujo_punto(wd13, x, y, "Black");
```

```
        break;
case 3: abrir_ventanas_esquema3(tf);

    fscanf(fp, "%s", texto); /*leo"\nDatos_caract.:..\n"*/
    fscanf(fp, "%lf ", &Lc);
    fscanf(fp, "%s", texto); /*leo"\nDatos:t, nc, ncd, Pc, Pcd, ...\n"*/

    u1d1 = u1d2 = u1d3 = u1d4 = u1d5 = tf;
    u1d6 = u1d7 = u1d8 = u1d9 = u1d10 = tf;

    v1d1=Lc;      v1d2=30000.;   v1d3=926.87;   v1d4=4000.;v1d5=2.;
    v1d6=926.87;  v1d7=100.;     v1d8=0.1;      v1d9=100.; v1d10=1.;

    for(cont=0.; cont<N_grupos; cont=cont+1.)
        {
        fscanf(fp, "%lf ", &t);
        fscanf(fp, "%lf ", &nc);      fscanf(fp, "%lf ", &ncd);
        fscanf(fp, "%lf ", &Pc);      fscanf(fp, "%lf ", &Pcd);
        fscanf(fp, "%lf ", &Tc);      fscanf(fp, "%lf ", &Tcd);
        fscanf(fp, "%lf ", &hves);        fscanf(fp, "%lf ", &hvss);
        fscanf(fp, "%lf ", &hvst);    fscanf(fp, "%lf ", &vves);
        fscanf(fp, "%lf ", &vvss);        fscanf(fp, "%lf ", &vvst);
        fscanf(fp, "%lf ", &Tvss);        fscanf(fp, "%lf ", &Tvst);
        fscanf(fp, "%lf ", &Tlsb);        fscanf(fp, "%lf ", &Qec);
        fscanf(fp, "%lf ", &Qes);     fscanf(fp, "%lf ", &Wvst);
        fscanf(fp, "%lf ", &Wvsc);        fscanf(fp, "%lf ", &Wb);
        fscanf(fp, "%lf ", &Wr);      fscanf(fp, "%lf ", &Tre);
        fscanf(fp, "%lf ", &Trs);     fscanf(fp, "%lf ", &kx);
        fscanf(fp, "%lf ", &R);

        cambio_coordenadas(t, nc, u1d1, v1d1);   dibujo_punto(wd31, x, y, "Black");
        cambio_coordenadas(t, ncd, u1d1, v1d1);  dibujo_punto(wd31, x, y, "Yellow");
        cambio_coordenadas(t, Pc, u1d2, v1d2);   dibujo_punto(wd32, x, y, "Black");
        cambio_coordenadas(t, Pcd, u1d2, v1d2);  dibujo_punto(wd32, x, y, "Yellow");
        cambio_coordenadas(t, Tc, u1d3, v1d3);   dibujo_punto(wd33, x, y, "Black");
        cambio_coordenadas(t, Tcd, u1d3, v1d3);  dibujo_punto(wd33, x, y, "Yellow");
        cambio_coordenadas(t, hves, u1d4, v1d4); dibujo_punto(wd34, x, y, "Black");
        cambio_coordenadas(t, hvss, u1d4, v1d4); dibujo_punto(wd34, x, y, "White");
        cambio_coordenadas(t, hvst, u1d4, v1d4); dibujo_punto(wd34, x, y, "Yellow");
        cambio_coordenadas(t, vves, u1d5, v1d5); dibujo_punto(wd35, x, y, "Black");
        cambio_coordenadas(t, vvss, u1d5, v1d5); dibujo_punto(wd35, x, y, "White");
        cambio_coordenadas(t, vvst, u1d5, v1d5); dibujo_punto(wd35, x, y, "Yellow");
        cambio_coordenadas(t, Tvss, u1d6, v1d6);dibujo_punto(wd36, x, y, "White");
        cambio_coordenadas(t, Tvst, u1d6, v1d6); dibujo_punto(wd36, x, y, "Yellow");
```



```
        cambio_coordenadas(t, Qec, u1d4, v1d4); dibujo_punto(wd14, x, y, "Black");
        cambio_coordenadas(t, hlec, u1d5, v1d5); dibujo_punto(wd15, x, y, "Black");
        cambio_coordenadas(t, Wlec, u1d6, v1d6);    dibujo_punto(wd16, x, y, "Black");
        cambio_coordenadas(t, Wvsc, u1d6, v1d6);    dibujo_punto(wd16, x, y, "Black");
        }
    break;
case 2: abrir_ventanas_esquema2(tf);
    fscanf(fp, "%s", texto);   /*leo"\nDatos_caract.:..\n"*/
    fscanf(fp, "%lf ", &Lc);
    fscanf(fp, "%s", texto);   /*leo"\nDatos:t, nc, Pves, Pvss, ...\n"*/

    u1d1 = u1d2 = u1d3 = u1d4 = u1d5 = u1d6 = u1d7 = u1d8 = tf;
    v1d1=Lc;      v1d2=50000.;   v1d3=926.87;   v1d4=4000.;
    v1d5=1.;      v1d6=10.;      v1d7=100.;     v1d8=0.1;

    for(cont=0.; cont<N_grupos; cont=cont+1.)
        {
        fscanf(fp, "%lf ", &t);
        fscanf(fp, "%lf ", &nc);      fscanf(fp, "%lf ", &Pves);
        fscanf(fp, "%lf ", &Pvss);        fscanf(fp, "%lf ", &Tves);
        fscanf(fp, "%lf ", &Tvss);        fscanf(fp, "%lf ", &hves);
        fscanf(fp, "%lf ", &hvss);        fscanf(fp, "%lf ", &hlec);
        fscanf(fp, "%lf ", &vves);        fscanf(fp, "%lf ", &vvss);
        fscanf(fp, "%lf ", &sves);    fscanf(fp, "%lf ", &svss);
        fscanf(fp, "%lf ", &Qec);     fscanf(fp, "%lf ", &Qes);
        fscanf(fp, "%lf ", &Wlec);        fscanf(fp, "%lf ", &Wvsc);

        cambio_coordenadas(t, nc, u1d1, v1d1);   dibujo_punto(wd21, x, y, "Black");
        cambio_coordenadas(t, Pves, u1d2, v1d2);dibujo_punto(wd22, x, y, "Black");
        cambio_coordenadas(t, Pvss, u1d2, v1d2);dibujo_punto(wd22, x, y, "Yellow");
        cambio_coordenadas(t, Tves, u1d3, v1d3);dibujo_punto(wd23, x, y, "Black");
        cambio_coordenadas(t, Tvss, u1d3, v1d3);dibujo_punto(wd23, x, y, "Yellow");
        cambio_coordenadas(t, hves, u1d4, v1d4);dibujo_punto(wd24, x, y, "Black");
        cambio_coordenadas(t, hvss, u1d4, v1d4);dibujo_punto(wd24, x, y, "Yellow");
        cambio_coordenadas(t, hlec, u1d4, v1d4);dibujo_punto(wd24, x, y, "White");
        cambio_coordenadas(t, vves, u1d5, v1d5);dibujo_punto(wd25, x, y, "Black");
        cambio_coordenadas(t, vvss, u1d5, v1d5);dibujo_punto(wd25, x, y, "Yellow");
        cambio_coordenadas(t, sves, u1d6, v1d6);dibujo_punto(wd26, x, y, "Black");
        cambio_coordenadas(t, svss, u1d6, v1d6);dibujo_punto(wd26, x, y, "Yellow");
        cambio_coordenadas(t, Qec, u1d7, v1d7);dibujo_punto(wd27, x, y, "White");
        cambio_coordenadas(t, Qes, u1d7, v1d7);dibujo_punto(wd27, x, y, "Black");
        cambio_coordenadas(t, Wlec, u1d8, v1d8);    dibujo_punto(wd28, x, y, "White");
        cambio_coordenadas(t, Wvsc, u1d8, v1d8);    dibujo_punto(wd28, x, y, "Black");
        }
```

```
            cambio_coordenadas(t, Tlsb, u1d6, v1d6); dibujo_punto(wd36, x, y, "Black");
            cambio_coordenadas(t, Qec, u1d7, v1d7); dibujo_punto(wd37, x, y, "Black");
            cambio_coordenadas(t, Qes, u1d7, v1d7); dibujo_punto(wd37, x, y, "White");
            cambio_coordenadas(t, Wvst, u1d8, v1d8);    dibujo_punto(wd38, x, y, "White");
            cambio_coordenadas(t, Wvsc, u1d8, v1d8);    dibujo_punto(wd38, x, y, "Black");
            cambio_coordenadas(t, Wb, u1d8, v1d8); dibujo_punto(wd38, x, y, "Red");
            cambio_coordenadas(t, Wr, u1d8, v1d8); dibujo_punto(wd38, x, y, "Yellow");
            cambio_coordenadas(t, Tre, u1d9, v1d9); dibujo_punto(wd39, x, y, "White");
            cambio_coordenadas(t, Trs, u1d9, v1d9); dibujo_punto(wd39, x, y, "Black");
            cambio_coordenadas(t, kx, u1d10, v1d10);    dibujo_punto(wd310, x, y, "Black");
            cambio_coordenadas(t, R, u1d10, v1d10); dibujo_punto(wd310, x, y, "Blue");
            }
        break;
    default:break;
    }
}
```

quitx.c

```
/* quitx.c */          /* CODIGO PARA CERRAR X */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>

#include "windows.h"
/*=========================================*/
extern Display   *el_display;
extern Cursor    el_cursor;
extern Window    w;
extern GC        el_GC;
/*=========================================*/
void quitX(void)     /* cierra la conexion con el servidor X  */
    {
    limpia_ventanas_GCs();
    limpiar_cursores();
    XFlush(el_display);
    XCloseDisplay( el_display );
    }
/*=========================================*/
```

```
void limpia_ventanas_GCs(void)
    {
    XFreeGC(el_display,el_GC);

    XDestroySubwindows(el_display, w);
    XDestroyWindow(el_display, w);
    XFlush(el_display);
}
```

raton.c

```
/** raton.c **/
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>

#include "windows.h"
/*=========================================*/
extern Display *el_display;

extern Window  w, w0, w1, w2, w3, w4;
extern Window  wmm, ws, wm, wc, we;
extern Window  wp, wpc, wpk, wp00, wp01, wp10, wp11, wp20, wp21, wp30, wp31, wp40, wp41;
extern Window          wp50, wp51, wp60, wp61, wp70, wp71, wp02, wp03, wp12, wp13;
extern Window          wp22, wp23, wp32, wp33, wp42, wp43, wp52, wp53, wp62, wp63;
extern Window  wg, wgsg, wgng, wgc, wge, wgg, wgg0, wgg1, wgg2, wgg3, wggb, wget, wgeb;
extern Window  wcn, wcn1, wcn2, wcn3, wcn4;
extern Window  wmt, wmtk, wmtc, wmta, wmt00, wmt01, wmtat, wmtab;
extern Window  wa, wa0, wa1, wa2, wa3, wam, wbam;
/*=========================================*/
int  Scheduler::raton_menu_inicial(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->window == w)      op=10;
    if(theEvent->window == w0)     op=0;
    if(theEvent->window == w1)     op=1;
    if(theEvent->window == w2)     op=2;
```

```
    if(theEvent->window == w3)      op=3;
    if(theEvent->window == w4)      op=4;
    return(op);
    }
/*=========================================*/
int Scheduler::raton_menu_modo(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->Rwindow == ws)             op=1;
    else if(theEvent->window == wm)         op=2;
    else if(theEvent->window == wc)     op=0;
    else if(theEvent->window == we)     op=3;
    else                                    op=10;

    return(op);
    }
/*=========================================*/
int Scheduler::raton_menu_parametros(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->window == wpk) op=1;
    else if(theEvent->window == wpc) op=0;

    else if(theEvent->window == wp00)       { t0=t0-1.;  op=10; }
    else if(theEvent->window == wp01)       { t0=t0+1.;  op=10; }
    else if(theEvent->window == wp10)       { tf=tf-100.; op=10; }
    else if(theEvent->window == wp11)       { tf=tf+100.; op=10; }
    else if(theEvent->window == wp20)       { h=h-0.01; op=10; }
    else if(theEvent->window == wp21)       { h=h+0.01; op=10; }
    else if(theEvent->window == wp30)       { P_caldera=P_caldera-100.; op=10; }
    else if(theEvent->window == wp31)       { P_caldera=P_caldera+100.; op=10; }
    else if(theEvent->window == wp40)       { P_condensador=P_condensador-1.; op=10; }
    else if(theEvent->window == wp41)       { P_condensador=P_condensador+1.; op=10; }
    else if(theEvent->window == wp50)       { W_refrigerante=W_refrigerante-0.001; op=10; }
    else if(theEvent->window == wp51)       { W_refrigerante=W_refrigerante+0.001; op=10; }
    else if(theEvent->window == wp60)       { T_refrig_in=T_refrig_in-1.; op=10; }
    else if(theEvent->window == wp61)       { T_refrig_in=T_refrig_in+1.; op=10; }
    else if(theEvent->window == wp70)       { T_refrig_out=T_refrig_out-1.; op=10; }
    else if(theEvent->window == wp71)       { T_refrig_out=T_refrig_out+1.; op=10; }
    else if(theEvent->window == wp02)       { m_refrigerante=m_refrigerante-0.1;op=10; }
    else if(theEvent->window == wp03)       { m_refrigerante=m_refrigerante+0.1;op=10; }
    else if(theEvent->window == wp12)       { W_bomba=W_bomba-0.001; op=10; }
    else if(theEvent->window == wp13)       { W_bomba=W_bomba+0.001; op=10; }
    else if(theEvent->window == wp22)       { W_valvula1=W_valvula1-0.001; op=10; }
    else if(theEvent->window == wp23)       { W_valvula1=W_valvula1+0.001; op=10; }
    else if(theEvent->window == wp32)       { kx_turbina=kx_turbina-0.01; op=10; }
    else if(theEvent->window == wp33)       { kx_turbina=kx_turbina+0.01; op=10; }
    else if(theEvent->window == wp42)       { h_supercal=h_supercal-100.; op=10; }
    else if(theEvent->window == wp43)       { h_supercal=h_supercal+100.; op=10; }
    else if(theEvent->window == wp52)       { m_supercal=m_supercal-0.1; op=10; }
    else if(theEvent->window == wp53)       { m_supercal=m_supercal+0.1; op=10; }
    else if(theEvent->window == wp62)       { k_supercal=k_supercal-0.5; op=10; }
    else if(theEvent->window == wp63)       { k_supercal=k_supercal+0.5; op=10; }
    else                                op=10;

    return(op);
    }
/*=========================================*/
int Scheduler::raton_menu_mostrar(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->window == wmtk)        op=1;
    else if(theEvent->window == wmtc)   op=0;
    else if(theEvent->window == wmta)   op=2;
    else if(theEvent->window == wmt00) op=3;
    else if(theEvent->Rwindow == wmt01)     op=4;
    else op=10;

    return(op);
    }
/*=========================================*/
int Scheduler::raton_menu_grabacion(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->window == wgc)             op=0;
    else if(theEvent->window == wgsg)       op=1;
    else if(theEvent->window == wgng)   op=2;
    else if(theEvent->window == wge)        op=3;
    else                                op=10;

    return(op);
    }
/*=========================================*/
int Scheduler::raton_set_grabacion(XButtonEvent *theEvent)
```

```
    {
    int op;

    if(theEvent->window == wggb)        op=0;
    else if(theEvent->window == wgg0)   { hit=hit-1.; op=1; }
    else if(theEvent->window == wgg1)   { hit=hit+1.; op=1; }
    else if(theEvent->window == wgg2)   { num_fich=num_fich-1; op=1; }
    else if(theEvent->window == wgg3)   { num_fich=num_fich+1; op=1; }
    else                                op=1;

    return(op);
    }
/*=========================================*/
int Scheduler::raton_control(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->window == wcn1)        op=1;
    else if(theEvent->window == wcn2)   op=2;
    else if(theEvent->window == wcn3)   op=3;
    else if(theEvent->window == wcn4)   op=0;
    else op=10;

    return(op);
    }
/*=========================================*/
int Scheduler::teclado_control(XKeyEvent *theEvent)
    {
    int op;

    if(theEvent->window == wcn1)        op=1;
    else if(theEvent->window == wcn2)   op=2;
    else if(theEvent->window == wcn3)   op=3;
    else if(theEvent->window == wcn4)   op=0;
    else        op=10;
    return(op);
}
/*=========================================*/
int Scheduler::raton_seguir(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->window == wcn3)        op=0;
    else if(theEvent->window == wcn4)   op=1;
    else                                op=2;
    return(op);
    }
/*=========================================*/
int raton_ayuda_modo(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->window == wbam)  op=0;
    else                          op=10;
    return(op);
    }
/*=========================================*/
int raton_ayuda_grabacion(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->window == wgeb)      op=0;
    else                          op=10;
    return(op);
    }
/*=========================================*/
int raton_ayuda_mostrar(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->window == wmtab) op=0;
    else                          op=10;
    return(op);
    }
/*=========================================*/
int raton_ayuda(XButtonEvent *theEvent)
    {
    int op;

    if(theEvent->window == wa0)         op=0;
    else if(theEvent->window == wa1)    op=1;
    else if(theEvent->window == wa2)    op=2;
    else if(theEvent->window == wa3)    op=3;
    else                                op=10;
    return(op);
}
```

```
    /*8*/   { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 151.87 },
    /*9*/   { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 179.92 },
    /*10*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 198.33 },
    /*11*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200. ,223.99 },
    /*12*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 242.60 },
    /*13*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 257.47 },
    /*14*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 275.62 },
    /*15*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 295.04 },
    /*16*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 300., 311.03 },
    /*17*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 300., 324.71 },
    /*18*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 300., 336.70 },
    /*19*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 300., 347.39 },
    /*20*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250.,300.,350.,357.04
},
    /*21*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250.,300.,350.,365.80
},
    /*22*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200.,250.,300.,350.,373.77
},
    /*23*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 300., 350., 400.,
                                                    600., 800., 1000.},
    /*24*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 300., 350., 400.,
                                                    600., 800., 1000.},
    /*25*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 300., 350., 400.,
                                                    600., 800., 1000.},
    /*26*/  { 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 300., 350., 400.,
                                                    600., 800., 1000.}
        };
    /*-------------------------------------------------------------------*/
    double h_l[27][22]= {
    /*0*/   { -0.04, -333.4 },
    /*1*/   { -0.04, 42.0, 73.4 },
    /*2*/   { -0.04, 42.0, 83.8, 121.3 },
    /*3*/   { -0.03, 42.0, 83.8, 125.7, 167.5, 168.8 },
    /*4*/   { -0.02, 42.0, 83.9, 125.7, 167.5, 209.3, 251.2, 251.5 },
    /*5*/   { -0.001, 42.0, 83.9, 125.7, 167.5, 209.4, 251.2, 293.0, 317.6 },
    /*6*/   { 0.03, 42.1, 83.9, 125.7, 167.6, 209.4, 251.2, 293.0, 334.9, 376.9, 384.4 },
    /*7*/   { 0.06, 21.1, 42.1, 63.0, 83.9, 125.8, 167.6, 209.4, 251.2, 293.1, 335.0, 377.0, 417.5 },
    /*8*/   { 0.5, 42.5, 84.3, 126.1, 167.9, 209.7, 251.6, 293.4, 335.3, 377.3, 419.4, 525.3, 632.3, 640.4 },
    /*9*/   { 1.0, 43.0, 84.8, 126.6, 168.4, 210.2, 252.0, 293.8, 335.7, 377.7, 419.7, 525.6, 632.6, 741.3, 762.9
},
    /*10*/  { 1.5, 43.4, 85.2, 127.0, 168.8, 210.6, 252.4, 294.2, 336.1, 378.0, 420.1, 525.9, 632.9, 741.5,844.9
},
    /*11*/  2.5, 44.4, 86.2, 127.9, 169.7, 211.5, 253.2, 295.0, 336.9, 378.8, 420.9, 526.6, 633.6, 742.1, 852.8,
                                                    962.0 },
    /*12*/  { 3.5, 45.4, 87.1, 128.9, 170.6, 212.3, 254.1, 295.8, 337.7, 379.6, 421.6, 527.3, 634.2, 742.6, 853.2,
                                                    1049.0 },
    /*13*/  { 4.5, 46.4, 88.1, 129.8, 171.5, 213.2, 254.9, 296.7, 338.5, 380.4, 422.4, 528.0, 634.8, 743.1, 853.6,
                                                    1085.0, 1121.0 },
    /*14*/  { 6.1, 47.8, 89.5, 131.1, 172.8, 214.5, 256.2, 297.9, 339.7, 381.5, 423.5, 529.1, 635.7, 743.9, 854.2,
                                                    1085.0, 1213.0 },
    /*15*/  { 8.1, 49.8, 91.3, 132.9, 174.6, 216.2, 257.9, 299.5, 341.3, 383.1, 425.0, 530.4, 637.0, 745.0, 855.1,
                                                    1085.0, 1316.0 },
    /*16*/  { 10.1, 51.7, 93.2, 134.8, 176.3, 217.9, 259.5, 301.2, 342.9, 384.6, 426.5, 531.8, 638.2,
746.1,855.9,
                                                    1085.0, 1342.0, 1407.0 },
    /*17*/  { 12.1, 53.6, 95.1, 136.6, 178.1, 219.6, 261.2, 302.8, 344.4, 386.2, 428.0, 533.2, 639.5, 747.2,
856.8,
                                                    1085.0, 1340.0, 1490.0 },
    /*18*/  { 14.1, 55.5, 96.9, 138.4, 179.9, 221.4, 262.9, 304.4, 346.0, 387.7, 429.5, 534.6, 640.8, 748.3,
                                                    857.7, 1085.0, 1338.0, 1570.0 },
    /*19*/{ 16.1, 57.5, 98.8, 140.2, 181.6, 223.1, 264.6, 306.1, 347.6, 389.3, 431.0, 536.0, 642.0, 749.4,
                                                    858.6, 1085.0, 1336.0, 1649.0 },
    /*20*/  { 18.1, 59.4, 100.6, 142.0, 183.4, 224.8, 266.2, 307.7, 349.2, 390.8, 432.6, 537.4, 643.3, 750.5,
                                                    859.5, 1086.0, 1334.0, 1732.0 },
    /*21*/  { 20.1, 61.3, 102.5, 143.8, 185.1, 226.5, 267.9, 309.3, 350.8, 392.4, 434.1, 538.8, 644.6, 751.6,
                                                    860.4, 1086.0, 1333.0, 1645.0, 1826.0 },
    /*22*/  { 22.1, 63.2, 104.3, 145.6, 186.9, 228.2, 269.6, 311.0, 352.4, 393.9, 435.6, 540.2, 645.9, 752.8,
                                                    861.3, 1086.0, 1332.0, 1635.0, 2012.0 },
    /*23*/  { 29.9, 70.8, 111.7, 152.7, 193.9, 235.0, 276.2, 317.5, 358.8, 400.2, 441.6, 545.9, 651.0, 757.3,
                                                    865.2, 1087.0, 1327.0, 1608.0, 2150.0, 3443.0, 4016.0, 4553.0 },
    /*24*/  { 49.2, 89.4, 129.9, 170.5, 211.2, 252.0, 292.9, 333.8, 374.7, 415.7, 456.8, 560.1, 664.2, 769.2,
                                                    875.3, 1093.0, 1323.0, 1575.0, 1874.0, 3247.0, 3919.0, 4495.0 },
    /*25*/  {68.0, 107.7, 147.7, 188.0, 228.4, 268.9, 309.4, 350.0, 390.6, 431.3, 472.1, 574.5, 677.6, 781.4,
                                                    886.2, 1100.0, 1323.0, 1560.0, 1822.0, 3063.0, 3826.0, 4439.0 },
    /*26*/  {95.4, 134.6, 174.1, 213.9, 253.9, 293.9, 334.0, 374.2, 414.4, 456.6, 495.0, 596.3, 698.1, 800.0,
                                                    903.6, 1112.0, 1328.0, 1553.0, 1790.0, 2863.0, 3704.0, 4362.0 }
        };
    /*-------------------------------------------------------------------*/
    double v_l[27][22]= {
    /*0*/   { 0.0010002, 0.0010908 },
    /*1*/   { 0.0010002, 0.0010003, 0.0010013 },
    /*2*/   { 0.0010002, 0.0010003, 0.0010018, 0.0010041 },
    /*3*/   { 0.0010002, 0.0010003, 0.0010018, 0.0010044, 0.0010079, 0.001008 },
    /*4*/   { 0.0010002, 0.0010003, 0.0010018, 0.0010044, 0.0010079, 0.0010122, 0.0010171, 0.0010172 },
    /*5*/   { 0.0010002, 0.0010003, 0.0010018, 0.0010044, 0.0010079, 0.0010121, 0.0010171, 0.0010228,
                                                    0.0010264 },
    /*6*/   { 0.0010002, 0.0010003, 0.0010018, 0.0010044, 0.0010079, 0.0010121, 0.0010171, 0.0010227,
                                                    0.0010290, 0.0010359, 0.0010372 },
```

```
/*7*/   { 0.0010002, 0.0010003, 0.0010018, 0.0010044, 0.0010079, 0.0010121, 0.0010171, 0.0010227,
          0.0010290, 0.0010359, 0.0010431 },
/*8*/   { 0.0010000, 0.0010001, 0.0010016, 0.0010042, 0.0010077, 0.0010119, 0.0010169, 0.0010225,
          0.0010288, 0.0010357, 0.0010432, 0.0010647, 0.0010904, 0.0010925 },
/*9*/   { 0.0009997, 0.0009998, 0.0010014, 0.0010040, 0.0010075, 0.0010117, 0.0010167, 0.0010223,
          0.0010286, 0.0010355, 0.0010430, 0.0010644, 0.0010901, 0.0011206, 0.0011272 },
/*10*/  { 0.0009995, 0.0009996, 0.0010011, 0.0010037, 0.0010072, 0.0010115, 0.0010165, 0.0010221,
          0.0010283, 0.0010352, 0.0010427, 0.0010641, 0.0010898, 0.0011201, 0.0011538 },
/*11*/  { 0.0009990, 0.0009991, 0.0010007, 0.0010033, 0.0010068, 0.0010110, 0.0010160, 0.0010216,
          0.0010279, 0.0010347, 0.0010422, 0.0010636, 0.0010891, 0.0011193, 0.0011554, 0.0011973 },
/*12*/  { 0.0009984, 0.0009986, 0.0010002, 0.0010028, 0.0010063, 0.0010106, 0.0010156, 0.0010212,
          0.0010274, 0.0010342, 0.0010417, 0.0010630, 0.0010884, 0.0011185, 0.0011544, 0.0012348 },
/*13*/  { 0.0009979, 0.0009982, 0.0009998, 0.0010024, 0.0010059, 0.0010102, 0.0010151, 0.0010207,
          0.0010269, 0.0010338, 0.0010412, 0.0010624, 0.0010877, 0.0011177, 0.0011534, 0.0012505,
          0.0012694 },
/*14*/  { 0.0009972, 0.0009975, 0.0009991, 0.0010017, 0.0010052, 0.0010095, 0.0010144, 0.0010200,
          0.0010262, 0.0010330, 0.0010404, 0.0010616, 0.0010868, 0.0011165, 0.0011519, 0.0012478,
          0.0013190 },
/*15*/  { 0.0009962, 0.0009965, 0.0009982, 0.0010009, 0.0010044, 0.0010086, 0.0010136, 0.0010191,
          0.0010253, 0.0010321, 0.0010394, 0.0010605, 0.0010855, 0.0011150, 0.0011500, 0.0012443,
          0.0013843 },
/*16*/  { 0.0009952, 0.0009956, 0.0009973, 0.0010000, 0.0010035, 0.0010078, 0.0010127, 0.0010182,
          0.0010244, 0.0010311, 0.0010384, 0.0010593, 0.0010842, 0.0011134, 0.0011481, 0.0012409,
          0.0013975, 0.0014522 },
/*17*/  { 0.0009942, 0.0009947, 0.0009964, 0.0009991, 0.0010026, 0.0010069, 0.0010118, 0.0010173,
          0.0010235, 0.0010302, 0.0010375, 0.0010582, 0.0010829, 0.0011119, 0.0011462, 0.0012375,
          0.0013892, 0.0015259 },
/*18*/  { 0.0009933, 0.0009938, 0.0009955, 0.0009983, 0.0010018, 0.0010060, 0.0010109, 0.0010164,
          0.0010226, 0.0010292, 0.0010365, 0.0010571, 0.0010816, 0.0011104, 0.0011443, 0.0012343,
          0.0013814, 0.0016096 },
/*19*/  { 0.0009923, 0.0009929, 0.0009946, 0.0009974, 0.0010009, 0.0010052, 0.0010101, 0.0010156,
          0.0010217, 0.0010283, 0.0010355, 0.0010561, 0.0010804, 0.0011089, 0.0011425, 0.0012311,
          0.0013740, 0.0017099 },
/*20*/  { 0.0009913, 0.0009920, 0.0009938, 0.0009965, 0.0010001, 0.0010043, 0.0010092, 0.0010147,
          0.0010208, 0.0010274, 0.0010346, 0.0010550, 0.0010791, 0.0011074, 0.0011407, 0.0012281,
          0.0013671, 0.0017028, 0.0018399 },
/*21*/  { 0.0009904, 0.0009911, 0.0009929, 0.0009957, 0.0009992, 0.0010035, 0.0010084, 0.0010138,
          0.0010199, 0.0010265, 0.0010336, 0.0010539, 0.0010779, 0.0011059, 0.0011389, 0.0012251,
          0.0013605, 0.0016645, 0.0020360 },
/*22*/  { 0.0009894, 0.0009902, 0.0009920, 0.0009948, 0.0009984, 0.0010026, 0.0010075, 0.0010130,
          0.0010190, 0.0010256, 0.0010327, 0.0010529, 0.0010767, 0.0011045, 0.0011371, 0.0012221,
          0.0013542, 0.0016343, 0.0027011 },
/*23*/  { 0.0009857, 0.0009866, 0.0009887, 0.0009915, 0.0009951, 0.0009993, 0.0010042, 0.0010096,
          0.0010155, 0.0010220, 0.0010290, 0.0010487, 0.0010719, 0.0010989, 0.0011303, 0.0012110,
          0.0013317, 0.0015522, 0.0027929, 0.0114310, 0.0156450, 0.0192810 },
/*24*/  { 0.0009767, 0.0009782, 0.0009805, 0.0009835, 0.0009872, 0.0009914, 0.0009962, 0.0010014,
          0.0010072, 0.0010134, 0.0010201, 0.0010389, 0.0010608, 0.0010859, 0.0011148, 0.0011869,
          0.0012876, 0.0014422, 0.0017301, 0.0060981, 0.0090834, 0.0114790 },
/*25*/  { 0.0009683, 0.0009702, 0.0009728, 0.0009759, 0.0009797, 0.0009839, 0.0009886, 0.0009938,
          0.0009994, 0.0010054, 0.0010118, 0.0010298, 0.0010505, 0.0010741, 0.0011010, 0.0011665,
          0.0012540, 0.0013774, 0.0015667, 0.0039725, 0.0063170, 0.0081534 },
/*26*/  { 0.0009567, 0.0009590, 0.0009619, 0.0009653, 0.0009691, 0.0009733, 0.0009780, 0.0009830,
          0.0009884, 0.0009942, 0.0010003, 0.0010173, 0.0010365, 0.0010582, 0.0010826, 0.0011406,
          0.0012148, 0.0013120, 0.0014439, 0.0026743, 0.0043285, 0.0056943 }
        };
/*-------------------------------------------------------------------------*/
double *arrT_l=&T_l[0][0];
double *arrh_l=&h_l[0][0];
double *arrv_l=&v_l[0][0];
int orden_interp_liq=3;
/*-------------------------------------------------------------------------*/
double e_liq[4][22]={
    { -0.0392145, 40.1942, 79.919, 119.179, 158.843, 199.676, 241.508,
    282.052, 322.325, 364.262, 418.991, 524.901, 631.999, 740.75, 851.754,
    1085.28, 1353.82, 1932.28, 3358., 3706., 4159., 4640.5 },
    { 0.00102177, 0.00116187, 0.0013436, 0.00158071, 0.00177723, 0.00185452,
    0.00183206, 0.00194509, 0.00209185, 0.00208397, 0.000750613, 0.000688171,
    0.0006182228, 0.000523101, 0.000403565, -8.60582e-5, -0.00138597,
    -0.0179914, -0.0614895, -0.00738238, -0.00457381, -0.00288024 },
    { -8.02481e-10, -4.79947e-9, -9.51608e-9, -1.54134e-8, -2.03459e-8,
    -2.2542e-8, -2.24763e-8, -2.54558e-8, -2.9237e-8, -2.98682e-8,
    1.49423e-10, 3.76654e-10, 5.89228e-10, 1.1276e-9, 1.54434e-9,
    5.92508e-9, 1.90047e-8, 2.79211e-7, 8.14286e-7, -6.14286e-8, -9.28571e-9,
    -2.14286e-9 },
    { 1.28784e-15, 2.63826e-14, 5.53498e-14, 9.14079e-14, 1.21623e-13,
    1.35125e-13, 1.34977e-13, 1.53251e-13, 1.76478e-13, 1.83819e-13,
    -5.42932e-16, -1.18711e-15, -1.61526e-15, -4.33679e-15, -3.9569e-15,
    -2.39363e-14, -7.73329e-14, -1.37379e-12, -3.5619e-12, 5.09524e-13,
    9.52381e-14, 3.09524e-14 } };
/*-------------------------------------------------------------------------*/
double T_dat[27]={ 0.001, 10., 20., 30., 40., 50., 60., 70., 80., 90., 100., 125., 150., 175., 200., 250., 300.,
            350., 400., 600., 800., 1000.};
/*************************************************************************/
/************************** LIQUIDO SATURADO **************************/
/*************************************************************************/
// h_sat_l[76] ==> kJ/kg      //
// s_sat_l[76] ==> kJ/(K*kg) //
// v_sat_l[76] ==> kg/m3 //
```

```
/*-------------------------------------------------------------------*/
double b_sat[4]={969.010985, -0.0607167, 4.67219e-6, -1.4474e-10};
double d_sat[4]={334.578144, 0.275225, -2.23808e-5, 6.19565e-10};
/*-------------------------------------------------------------------*/
double s_sat_l[76]={.0012, .0757, .1509, .2244, .2965, .3672, .4367, .5049, .5720,
        .6381, .7031, .7672, .8303, .8926, .9540, 1.0146, 1.0744, 1.1335, 1.1917, 1.2493,
        1.3062, 1.3624, 1.4179, 1.4728, 1.5271, 1.5807, 1.6338, 1.6864, 1.7384, 1.7899,
        1.8409, 1.8915, 1.9416, 1.9912, 2.0405, 2.0894, 2.1380, 2.1862, 2.2341, 2.2817,
        2.3290, 2.3761, 2.4230, 2.4696, 2.5161, 2.5623, 2.6084, 2.6544, 2.7002, 2.7460,
        2.7917, 2.8373, 2.8829, 2.9286, 2.9743, 3.0200, 3.0660, 3.1121, 3.1585, 3.2052,
        3.2523, 3.3000, 3.3483, 3.3973, 3.4473, 3.4984, 3.5507, 3.6045, 3.6601, 3.7176,
        3.7775, 3.8400, 3.9056, 3.9746, 4.0476, 4.4298};
/*-------------------------------------------------------------------*/
double v_sat_l[76]={.001, .001, .001, .001001, .001002, .001003, .001004, .001006,
        .001008, .001010, .001012, .001014, .001017, .001020, .001023, .001026,
        .001029, .001032, .001036, .001039, .001043, .001047, .001051, .001056,
        .001060, .001065, .001070, .001075, .001080, .001085, .001091, .001096,
        .001102, .001108, .001114, .001121, .001127, .001134, .001141, .001148,
        .001156, .001164, .001172, .001180, .001189, .001198, .001208, .001218,
        .001228, .001239, .001250, .001262, .001275, .001288, .001302, .001317,
        .001333, .001349, .001366, .001385, .001404, .001425, .001447, .001470,
        .001499, .001528, .001561, .001598, .001639, .001686, .001741, .001808,
        .001896, .002016, .002225, .00315 };
/*-------------------------------------------------------------------*/
double h_sat_l[76]={0.0, 21.0, 42.0, 63.0, 83.9, 104.8, 125.6, 146.5, 167.4, 188.3,
        209.1, 230.0, 250.9, 271.9, 292.8, 313.7, 334.7, 355.7, 376.7, 397.8, 418.9,
        440.0, 461.1, 482.3, 503.5, 524.8, 546.1, 567.4, 588.8, 610.3, 631.9, 653.5,
        675.2, 697.0, 718.8, 740.8, 762.8, 785.0, 807.2, 829.6, 852.1, 874.7, 897.5,
        920.4, 943.5, 966.7, 990.1, 1013.7, 1037.5, 1061.4, 1085.6, 1110.1, 1134.8,
        1159.8, 1185.1, 1210.7, 1236.6, 1263.0, 1289.8, 1317.1, 1344.9, 1373.3,
        1402.3, 1432.1, 1462.6, 1494.0, 1526.3, 1559.7, 1594.3, 1632.5, 1671.8,
        1713.9, 1761.6, 1817.8, 1890.1, 2099.3};
/*********************************************************************************/
/************************** VAPOR SUPERCALENTADO *************************/
/*********************************************************************************/
// P_sup[28]     ==> kPa       //
// TK_sup[28][18] ==> K          //
// h_sup[28][18] ==> kJ/kg       //
// s_sup[28][18] ==> kJ/(K*kg)   //
// v_sup[28][18] ==> m3/kg       //
/*-------------------------------------------------------------------*/
double P_sup[28]={ 1., 2., 4., 7., 10., 20., 40., 70., 101.32, 200., 400., 700., 1000.,
        2000., 3000., 4000., 6000., 8000., 10000., 15000., 20000., 30000., 40000.,
        50000., 60000., 70000., 80000., 100000.};
/*-------------------------------------------------------------------*/
double TK_sup[28][18]={
/*0*/     { 280.13, 350., 400., 450., 500., 550., 600., 650., 700., 750., 800.,
        850., 900., 950., 1000., 1050., 1100., 1150.} ,
/*1*/     { 290.65, 350., 400., 450., 500., 550., 600., 650., 700., 750., 800.,
        850., 900., 950., 1000., 1050., 1100., 1150.} ,
/*2*/     { 302.12, 350., 400., 450., 500., 550., 600., 650., 700., 750., 800.,
        850., 900., 950., 1000., 1050., 1100., 1150.} ,
/*3*/     { 311.98, 350., 400., 450., 500., 550., 600., 650., 700., 750., 800.,
        850., 900., 950., 1000., 1050., 1100., 1150.} ,
/*4*/     { 318.96, 350., 400., 450., 500., 550., 600., 650., 700., 750., 800.,
        850., 900., 950., 1000., 1050., 1100., 1150.} ,
/*5*/     { 333.22, 350., 400., 450., 500., 550., 600., 650., 700., 750., 800.,
        850., 900., 950., 1000., 1050., 1100., 1150.} ,
/*6*/     { 349.02, 350., 400., 450., 500., 550., 600., 650., 700., 750., 800.,
        850., 900., 950., 1000., 1050., 1100., 1150.} ,
/*7*/     { 362.87, 400., 450., 500., 550., 600., 650., 700., 750., 800., 850.,
        900., 950., 1000., 1050., 1100., 1150., 1200.} ,
/*8*/     { 373.14, 400., 450., 500., 550., 600., 650., 700., 750., 800., 850.,
        900., 950., 1000., 1050., 1100., 1150., 1200.} ,
/*9*/     { 393.38, 400., 450., 500., 550., 600., 650., 700., 750., 800., 850.,
        900., 950., 1000., 1050., 1100., 1150., 1200.} ,
/*10*/    { 416.78, 450., 500., 550., 600., 650., 700., 750., 800., 850., 900.,
        950., 1000., 1050., 1100., 1150., 1200.} ,
/*11*/    { 437.79, 450., 500., 550., 600., 650., 700., 750., 800., 850., 900.,
        950., 1000., 1050., 1100., 1150., 1200.} ,
/*12*/    { 453.06, 500., 550., 600., 650., 700., 750., 800., 850., 900., 950.,
        1000., 1050., 1100., 1150., 1200.} ,
/*13*/    { 485.57, 500., 550., 600., 650., 700., 750., 800., 850., 900., 950.,
        1000., 1050., 1100., 1150., 1200.} ,
/*14*/    { 507.05, 600., 650., 700., 750., 800., 900., 1000., 1100.} ,
/*15*/    { 523.55, 600., 650., 700., 750., 800., 900., 1000., 1100.} ,
/*16*/    { 548.79, 600., 650., 700., 750., 800., 900., 1000., 1100.} ,
/*17*/    { 568.22, 600., 650., 700., 750., 800., 900., 1000., 1100.} ,
/*18*/    { 584.22, 600., 650., 700., 750., 800., 900., 1000., 1100.} ,
/*19*/    { 615.24, 650., 700., 750., 800., 900., 1000., 1100.} ,
/*20*/    { 638.96, 650., 700., 750., 800., 900., 1000., 1100.} ,
/*21*/    { 700., 750., 800., 850., 900., 950., 1000., 1050., 1100.} ,
/*22*/    { 700., 750., 800., 850., 900., 950., 1000., 1050., 1100.} ,
/*23*/    { 700., 750., 800., 850., 900., 950., 1000., 1050., 1100.} ,
/*24*/    { 700., 750., 800., 850., 900., 950., 1000., 1050., 1100.} ,
/*25*/    { 700., 750., 800., 850., 900., 950., 1000., 1050., 1100.} ,
/*26*/    { 700., 750., 800., 850., 900., 950., 1000., 1050., 1100.} ,
/*27*/    { 700., 750., 800., 850., 900., 950., 1000., 1050., 1100.}
```

```
        };
/*-------------------------------------------------------------------*/
double v_sup[28][18]={
/*0*/   { 129.2, 161.5, 184.6, 207.7, 230.7, 253.8, 276.9, 300.0, 323.1, 346.1,
      369.2, 392.3, 415.4, 438.4, 461.5, 484.6, 507.7, 530.7 },
/*1*/   { 67., 80.73, 92.28, 103.8, 115.4, 126.9, 138.4, 150., 161.5, 173.1,
      184.6, 196.1, 207.7, 219.2, 230.8, 242.3, 253.8, 265.4 },
/*2*/   { 34.8, 40.35, 46.13, 51.91, 57.68, 63.45, 69.22, 74.99, 80.76, 86.53,
      92.3, 98.07, 103.8, 109.6, 115.4, 121.1, 126.9, 132.7 },
/*3*/   { 20.53, 23.04, 26.35, 29.66, 32.96, 36.25, 39.55, 42.85, 46.15, 49.45,
      52.74, 56.04, 59.34, 62.63, 65.93, 69.23, 72.52, 75.82 },
/*4*/   { 14.67, 16.12, 18.44, 20.75, 23.07, 25.38, 27.69, 29.99, 32.3, 34.61,
      36.92, 39.23, 41.54, 43.84, 46.15, 48.46, 50.77, 53.07 },
/*5*/   { 7.649, 8.044, 9.21, 10.37, 11.53, 12.68, 13.84, 14.99, 16.15, 17.3,
      18.46, 19.61, 20.77, 21.92, 23.07, 24.23, 25.38, 26.54 },
/*6*/   { 3.993, 4.005, 4.595, 5.179, 5.759, 6.339, 6.917, 7.495, 8.073, 8.65,
      9.228, 9.805, 10.38, 10.96, 11.54, 12.11, 12.69, 13.27 },
/*7*/   { 2.365, 2.617, 2.953, 3.287, 3.619, 3.95, 4.281, 4.612, 4.942, 5.272,
      5.602, 5.932, 6.262, 6.592, 6.922, 7.251, 7.581, 7.911 },
/*8*/   { 1.673, 1.802, 2.036, 2.268, 2.498, 2.727, 2.956, 3.185, 3.413, 3.641,
      3.87, 4.098, 4.326, 4.554, 4.781, 5.009, 5.237, 5.465 },
/*9*/   { 0.8857, 0.9024, 1.025, 1.144, 1.262, 1.379, 1.495, 1.612, 1.728,
      1.844, 1.959, 2.075, 2.191, 2.306, 2.422, 2.537, 2.653, 2.768 },
/*10*/  { 0.4625, 0.5053, 0.5671, 0.6273, 0.6866, 0.7455, 0.8040, 0.8624,
      0.9206, 0.9787, 1.037, 1.095, 1.153, 1.21, 1.268, 1.326, 1.384 },
/*11*/  { 0.2729, 0.2822, 0.3197, 0.3552, 0.3899, 0.424, 0.4579, 0.4915,
      0.525, .5584, .5917, .6249, .6581, 0.6912, 0.7244, 0.7575, 0.7905 },
/*12*/  { 0.1944, 0.2206, 0.2464, 0.2712, 0.2955, 0.3194, 0.3432, 0.3668,
      0.3902, 0.4137, 0.437, 0.4603, 0.4836, 0.5068, 0.53, 0.5532 },
/*13*/  { 0.09963, 0.1044, 0.1191, 0.1326, 0.1454, 0.1578, 0.1701, 0.1821,
      0.1941, 0.206, 0.2178, 0.2295, 0.2413, 0.253, 0.2646, 0.2763 },
/*14*/  { .06668, .08628, .09532, .104, .1124, .1206, .1367, 0.1526, 0.1684 },
/*15*/  { .04978, .06304, .07024, .07699, .08849, .08981, .1021, .1142, .1261},
/*16*/  { .03244, .03958, .04507, .04998, .05459, .05901, .0675, .07571, .08375 },
/*17*/  { .02352, .02759, .03239, .03643, .04011, .04359, .05018, .05648, .0626 },
/*18*/  { .01803, .02008, .02468, .02825, .03141, .03434, .03979, .04494, .04992},
/*19*/  { .01034, .01404, .01723, .01975, .02196, .02593, .02956, .03301 },
/*20*/  { .00584, .0079, .01156, .01386, .01575, .019, .02188, .02456 },
/*21*/  { .00543, .00786, .00951, .01087, .01208, .01318, .01421, .01518, .01612 },
/*22*/  { .0026, .00483, .0064, .0076, .00863, .00954, .01039, .01117, .01192 },
/*23*/  { .00203, .00323, .00459, .00567, .00658, .00738, .00811, .00878, .00941 },
/*24*/  { .00183, .00251, .0035, .00444, .00526, .00597, .00661, .0072, .00775 },
/*25*/  { .00172, .00217, .00287, .00364, .00435, .00498, .00556, .00609, .00658 },
/*26*/  { .00164, .00197, .00248, .0031, .00371, .00427, .00479, .00527, .00571 },
/*27*/  { .00153, .00176, .00207, .00247, .00291, .00335, .00377, .00416, .00453 }
        };
/*-------------------------------------------------------------------*/
double h_sup[28][18]={
/*0*/   { 2513.7, 2644.4, 2739.0, 2834.7, 2931.8, 3030.2, 3130.1, 3231.7, 3334.8,
      3439.6, 3546.1, 3654.3, 3764.3, 3876.0, 3989.5, 4104.7, 4221.7, 4340.5 },
/*1*/   { 2533.0, 2644.3, 2738.9, 2834.7, 2931.7, 3030.2, 3130.1, 3231.6, 3334.8,
      3439.6, 3546.1, 3654.3, 3764.3, 3876.0, 3989.5, 4104.7, 4221.7, 4340.5 },
/*2*/   { 2553.9, 2644.0, 2738.8, 2834.6, 2931.7, 3030.1, 3130.1, 3231.6, 3334.8,
      3439.6, 3546.1, 3654.3, 3764.3, 3876.0, 3989.5, 4104.7, 4221.7, 4340.5 },
/*3*/   { 2572.0, 2643.5, 2738.5, 2834.4, 2931.5, 3030.0, 3130.0, 3231.6, 3334.7,
      3439.5, 3546.0, 3654.3, 3764.3, 3876.0, 3989.5, 4104.7, 4221.7, 4340.5 },
/*4*/   { 2584.2, 2643.0, 2738.2, 2834.2, 2931.4, 3030.0, 3130.0, 3231.5, 3334.7,
      3439.5, 3546.0, 3654.3, 3764.2, 3876.0, 3989.5, 4104.7, 4221.7, 4340.4 },
/*5*/   { 2609.3, 2641.4, 2737.3, 2833.7, 2931.0, 3029.7, 3129.7, 3231.3, 3334.5,
      3439.4, 3545.9, 3654.2, 3764.2, 3875.9, 3989.4, 4104.7, 4221.7, 4340.4 },
/*6*/   { 2636.3, 2638.2, 2735.5, 2832.5, 2930.3, 3029.1, 3129.3, 3231.0, 3334.3,
      3439.2, 3545.7, 3654.0, 3764.0, 3875.8, 3989.3, 4104.6, 4221.6, 4340.3 },
/*7*/   { 2659.6, 2732.7, 2830.8, 2929.1, 3028.3, 3128.7, 3230.5, 3333.8, 3438.8,
      3545.4, 3653.8, 3763.8, 3875.6, 3989.1, 4104.4, 4221.5, 4340.2, 4460.7 },
/*8*/   { 2675.6, 2729.7, 2829.0, 2927.9, 3027.4, 3128.0, 3229.9, 3333.4, 3438.4,
      3545.1, 3653.5, 3763.6, 3875.4, 3989.0, 4104.3, 4221.3, 4340.1, 4460.6 },
/*9*/   { 2706.2, 2720.2, 2823.2, 2924.0, 3024.6, 3125.8, 3228.2, 33332.0, 3437.3,
      3544.2, 3652.7, 3762.9, 3874.8, 3988.5, 4103.8, 4220.9, 4339.7, 4460.3 },
/*10*/  { 2738.1, 2811.0, 2916.0, 3018.8, 3121.5, 3224.8, 3329.2, 3435.0, 3542.2,
      3651.1, 3761.5, 3873.6, 3987.4, 4102.9, 4220.1, 4339.0, 4459.6 },
/*11*/  { 2763.1, 2791.3, 2903.4, 3010.0, 3114.8, 3219.5, 3325.0, 3431.5, 3539.3,
      3648.6, 3759.4, 3871.8, 3985.8, 4101.5, 4218.9, 4337.9, 4458.6 },
/*12*/  { 2777.6, 2890.2, 3000.9, 3108.0, 3214.2, 3320.7, 3428.0, 3536.4, 3646.1,
      3757.3, 3870.0, 3984.3, 4100.1, 4217.6, 4336.8, 4457.6 },
/*13*/  { 2799.1, 2840.5, 2968.4, 3084.5, 3196.1, 3306.2, 3416.1, 3526.5, 3637.8,
      3750.2, 3863.9, 3979.0, 4095.5, 4213.5, 4333.1, 4454.2 },
/*14*/  { 2803.7, 3059.6, 3177.3, 3291.3, 3404.0, 3516.5, 3743.1, 3973.7, 4209.4 },
/*15*/  { 2801.0, 3033.0, 3157.8, 3276.1, 3391.6, 3506.3, 3735.8, 3968.3, 4205.3 },
/*16*/  { 2783.9, 2973.8, 3116.3, 3244.4, 3366.3, 3485.5, 3721.2, 3957.5, 4197.0 },
/*17*/  { 2757.5, 2904.1, 3071.1, 3210.9, 3340.0, 3464.0, 3706.2, 3946.6, 4188.7 },
/*18*/  { 2724.3, 2818.3, 3021.4, 3175.6, 3312.6, 3442.0, 3691.0, 3935.6, 4180.3 },
/*19*/  { 2610.1, 2868.5, 3077.4, 3239.6, 3384.3, 3652.0, 3907.7, 4159.4 },
/*20*/  { 2409.5, 2625.1, 2961.0, 3159.4, 3323.0, 3611.7, 3879.3, 4138.5 },
/*21*/  { 2633.3, 2973.5, 3189.4, 3367.6, 3528.0, 3678.1, 3821.6, 3960.6, 4096.5 },
/*22*/  { 2221.6, 2752.8, 3043.7, 3258.4, 3441.6, 3607.8, 3763.3, 3911.6, 4054.6 },
/*23*/  { 2074.8, 2535.1, 2893.2, 3147.2, 3354.5, 3537.3, 3705.1, 3862.8, 4013.2 },
/*24*/  { 2013.6, 2387.9, 2754.9, 3039.5, 3269.1, 3468.0, 3647.8, 3814.7, 3972.3 },
/*25*/  { 1977.8, 2301.9, 2644.3, 2941.8, 3188.5, 3401.4, 3592.2, 3767.9, 3932.5 },
```

```
/*26*/   { 1953.8, 2248.4, 2563.0, 2858.7, 3115.3, 3339.0, 3539.4, 3722.9, 3893.9 },
/*27*/   { 1923.7, 2186.1, 2461.2, 2736.7, 2995.5, 3230.7, 3444.2, 3640.0, 3821.8 }
         };
/*-------------------------------------------------------------------*/
double s_sup[28][18]={
/*0*/    { 8.9748, 9.3913, 9.6439, 9.8693, 10.0737, 10.2614, 10.4353, 10.5978, 10.7506, 10.8952, 11.0327,
    11.1639, 11.2896, 11.4104, 11.5268, 11.6392, 11.7481, 11.8536},
/*1*/    { 8.7228, 9.0710, 9.3238, 9.5493, 9.7538, 9.9414, 10.1153, 10.2779, 10.4307, 10.5753, 10.7128,
    10.8440, 10.9697, 11.0905, 11.2069, 11.3193, 11.4282, 11.5337},
/*2*/    { 8.4738, 8.7504, 9.0035, 9.2292, 9.4338, 9.6215, 9.7954, 9.9579, 10.1108, 10.2554, 10.3929,
    10.5241, 10.6498, 10.7706, 10.8870, 10.9994, 11.1083, 11.2138},
/*3*/    { 8.2750, 8.4911, 8.7447, 8.9707, 9.1753, 9.3631, 9.5371, 9.6996, 9.8525, 9.9971, 10.1346,
    10.2658, 10.3915, 10.5123, 10.6287, 10.7412, 10.8500, 10.9556},
/*4*/    { 8.1494, 8.3254, 8.5796, 8.8058, 9.0106, 9.1984, 9.3724, 9.5349, 9.6878, 9.8324, 9.9699, 10.1011,
    10.2269, 10.3477, 10.4641, 10.5765, 10.6854, 10.7910},
/*5*/    { 7.9077, 8.0019, 8.2579, 8.4849, 8.6901, 8.8781, 9.0522, 9.2148, 9.3678, 9.5124, 9.6499, 9.7812,
    9.9069, 10.0277, 10.1442, 10.2566, 10.3655, 10.4710},
/*6*/    { 7.6692, 7.6747, 7.9344, 8.1631, 8.3690, 8.5574, 8.7318, 8.8945, 9.0476, 9.1923, 9.3299, 9.4611,
    9.5869, 9.7077, 9.8242, 9.9366, 10.0455, 10.1511},
/*7*/    { 7.4789, 7.6707, 7.9019, 8.1090, 8.2980, 8.4727, 8.6357, 8.7889, 8.9337, 9.0713, 9.2027, 9.3284,
    9.4493, 9.5658, 9.6783, 9.7872, 9.8927, 9.9953 },
/*8*/    { 7.3541, 7.4942, 7.7281, 7.9365, 8.1261, 8.3012, 8.4644, 8.6177, 8.7627, 8.9004, 9.0317, 9.1576,
    9.2785, 9.3950, 9.5075, 9.6164, 9.7220, 9.8245 },
/*9*/    { 7.1263, 7.1616, 7.4044, 7.6168, 7.8085, 7.9847, 8.1486, 8.3024, 8.4477, 8.5856, 8.7172, 8.8432,
    8.9642, 9.0808, 9.1933, 9.3023, 9.4079, 9.5105 },
/*10*/   { 6.8951, 7.0634, 7.2848, 7.4808, 7.6594, 7.8248, 7.9795, 8.1255, 8.2639, 8.3959, 8.5221, 8.6433,
    8.7601, 8.8728, 8.9818, 9.0875, 9.1901 },
/*11*/   { 6.7072, 6.7709, 7.0073, 7.2104, 7.3928, 7.5605, 7.7168, 7.8637, 8.0029, 8.1354, 8.2621, 8.3836,
    8.5006, 8.6135, 8.7226, 8.8284, 8.9312 },
/*12*/   { 6.5857, 6.8223, 7.0333, 7.2198, 7.3898, 7.5476, 7.6956, 7.8356, 7.9686, 8.0957, 8.2175, 8.3348,
    8.4478, 8.5571, 8.6631, 8.7659 },
/*13*/   { 6.3401, 6.4242, 6.6683, 6.8704, 7.0491, 7.2122, 7.3639, 7.5064, 7.6413, 7.7698, 7.8928, 8.0108,
    8.1245, 8.2343, 8.3406, 8.4437 },
/*14*/   { 6.1861, 6.6516, 6.8402, 7.0091, 7.1646, 7.3098, 7.5767, 7.8196, 8.0442 },
/*15*/   { 6.0693, 6.4847, 6.6846, 6.8600, 7.0194, 7.1674, 7.4378, 7.6827, 7.9085 },
/*16*/   { 5.8884, 6.2201, 6.4486, 6.6385, 6.8067, 6.9605, 7.2382, 7.4872, 7.7154 },
/*17*/   { 5.7424, 5.9938, 6.2617, 6.4691, 6.6472, 6.8073, 7.0927, 7.3459, 7.5766 },
/*18*/   { 5.6133, 5.7722, 6.0983, 6.3270, 6.5162, 6.6833, 6.9767, 7.2344, 7.4676 },
/*19*/   { 5.3090, 5.7192, 6.0295, 6.2536, 6.4404, 6.7559, 7.0254, 7.2656 },
/*20*/   { 4.9265, 5.2616, 5.7622, 6.0363, 6.2477, 6.5881, 6.8702, 7.1172 },
/*21*/   { 5.1776, 5.6490, 5.9280, 6.1442, 6.3276, 6.4900, 6.6373, 6.7729, 6.8993 },
/*22*/   { 4.5363, 5.2720, 5.6483, 5.9089, 6.1185, 6.2982, 6.4578, 6.6025, 6.7357 },
/*23*/   { 4.2943, 4.9293, 5.3926, 5.7009, 5.9380, 6.1358, 6.3080, 6.4619, 6.6019 },
/*24*/   { 4.1795, 4.6954, 5.1697, 5.5152, 5.7779, 5.9930, 6.1776, 6.3405, 6.4872 },
/*25*/   { 4.1031, 4.5498, 4.9920, 5.3530, 5.6353, 5.8656, 6.0615, 6.2329, 6.3861 },
/*26*/   { 4.0448, 4.4510, 4.8571, 5.2159, 5.5094, 5.7514, 5.9571, 6.1362, 6.2953 },
/*27*/   { 3.9567, 4.3186, 4.6735, 5.0077, 5.3037, 5.5581, 5.7772, 5.9684, 6.1375 }
     };
/*-------------------------------------------------------------------*/
double *arrTK_sup=&TK_sup[0][0];
double *arrh_sup=&h_sup[0][0];
double *arrv_sup=&v_sup[0][0];
double *arrs_sup=&s_sup[0][0];
/*****************************************************************************/
/*********************************  VAPOR  ***********************************/
/*****************************************************************************/
// P_sat[76] ==> kPa      //
// T_sat[76] ==> C        //   z = zl+x*(zg-zl) = zl+x*zlg
// h_sat_v[76] ==> kJ/kg //   donde z = h, s, v
// s_sat_v[76] ==> kJ/(K*kg) //   l = sat_l
// v_sat_v[76] ==> m3/kg     //   g = sat_v
// h_sat_l[76] ==> kJ/kg  //
// s_sat_l[76] ==> kJ/(K*kg)  //
// v_sat_l[76] ==> m3/kg //
/*-------------------------------------------------------------------*/
int orden_interp_vap=3;
double x_dat[11]={ 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 };
/*-------------------------------------------------------------------*/
double c_vap[4][11]={
        { 334.578, 565.849, 797.12, 1028.39, 1259.66, 1490.93, 1722.2, 1953.48, 2184.75, 2416.02,
        2647.29 },
        { 0.275225, 0.253816, 0.232406, 0.210996, 0.189586, 0.168176, 0.146767, 0.125357,
        0.103947, 0.0825374, 0.0611276 },
        { -2.23808e-5, -2.07405e-5, -1.91003e-5, -1.746e-5, -1.58198e-5, -1.41795e-5, -1.25392e-5,
        -1.0899e-5, -9.25871e-6, -7.61845e-6, -5.97818e-6 },
        { 6.19565e-10, 5.68559e-10, 5.17552e-10, 4.66546e-10, 4.1554e-10, 3.64533e-10,
        3.13527e-10, 2.62521e-10, 2.11514e-10, 1.60508e-10, 1.09501e-10 } };
/*-------------------------------------------------------------------*/
double a_vap[4][11]={
        { 969.011, 871.939, 774.868, 677.796, 580.724, 483.653, 386.581, 289.51, 192.438, 95.3663,
        -1.70535 },
        { -0.0607167, -0.0537018, -0.0466869, -0.0396721, -0.0326572, -0.0256424, -0.0186275,
        -0.0116126, -0.00459776, 0.0024171, 0.00943197 },
        { 4.67219e-6, 4.1143e-6, 3.55641e-6, 2.99852e-6, 2.44063e-6, 1.88273e-6, 1.32484e-6,
        7.66951e-7, 2.0906e-7, -3.48832e-7, -9.06723e-7 },
        { -1.4474e-10, -1.25608e-10, -1.06476e-10, -8.73436e-11, -6.82115e-11, -4.90794e-11,
        -2.99473e-11, -1.08152e-11, 8.31691e-12, 2.7449e-11, 4.65811e-11 } };
```

teclado.c

```
/* teclado.c */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/keysym.h>
#include <X11/keysymdef.h>
#include <stdio.h>

#include "windows.h"
/*=========================================*/
extern Display *el_display;


int                 length;
int                 long_max_buffer = 64;
char                buffer_tecla[65];
KeySym              simbolo_tecla;
XComposeStatus      theComposeStatus;
/*=========================================*/
int procesa_tecla(XKeyEvent *theEvent)
    {
    int         length;
    int         long_max_buffer = 64;
    char        buffer_tecla[65];
    KeySym      simbolo_tecla;
    XComposeStatus  theComposeStatus;

    length=XLookupString(theEvent, buffer_tecla, long_max_buffer, &simbolo_tecla,
                                                    &theComposeStatus );
    if(theEvent->state & Mod1Mask)
        {
        switch(buffer_tecla[0])
            {
            case 'Q':
            case 'q':printf("META-Q hit\n"); return(0); break;
            }
        printf("META [%s]\n", buffer_tecla); return(1);
        }
    if( (simbolo_tecla>=' ') && (simbolo_tecla<='~') && (length>0) )
        {
        printf("ASCII key was hit: [%s]\n", buffer_tecla);
    if((buffer_tecla[0]=='q')||(buffer_tecla[0]=='Q'))
        { return(0); }
    }
else
    {
    switch(simbolo_tecla)
        {
        case XK_Return:         printf("Return\n"); break;
        case XK_BackSpace:      printf("BackSpace\n"); break;
        case XK_Escape:         printf("ESCAPE\n"); break;
        case XK_Delete:     printf("Delete\n"); break;
        case XK_Up:             printf("Up\n"); break;
        case XK_Down:       printf("Down\n"); break;
        case XK_Right:      printf("Right\n"); break;
        case XK_Left:       printf("Left\n"); break;
        case XK_R9 :        /* -- Sun 386i Mapping */
        case XK_Prior:      printf("Prior\n"); break;
        case XK_R15:
        case XK_Next:       printf("Next\n"); break;
        case XK_R13:        /* -- Sun 386i Mapping */
        case XK_End:        printf("End\n"); break;
        case XK_R7:             /* -- Sun 386i Mapping */
        case XK_Begin:      printf("Begin\n"); break;
        case XK_Insert:     printf("Insert\n"); break;
        case XK_Help:       printf("Help\n"); break;
        case XK_F1:             printf("F1\n"); break;
        case XK_F2:             printf("F2\n"); break;
        case XK_F3:             printf("F3\n"); break;
        case XK_F4:             printf("F4\n"); break;
        case XK_F5:             printf("F5\n"); break;
        case XK_F6:             printf("F6\n"); break;
        case XK_F7:             printf("F7\n"); break;
        case XK_F8:             printf("F8\n"); break;
        case XK_F9:             printf("F9\n"); break;
        case XK_F10:        printf("F10\n"); break;
        case XK_F11:        printf("F11\n"); break;
        case XK_F12:        printf("F12\n"); break;
        case XK_F13:        printf("F13\n"); break;
        case XK_F14:        printf("F14\n"); break;
        case XK_F15:        printf("F15\n"); break;
        case XK_KP_Enter:       printf("Keypad Enter\n"); break;
        case XK_KP_Equal: printf("Keypad Equal\n"); break;
        case XK_KP_Multiply:  printf("Keypad *\n"); break;
        case XK_KP_Add:         printf("Keypad +\n"); break;
```

```
            case XK_KP_Subtract:   printf("Keypad -\n"); break;
            case XK_KP_Decimal:    printf("Keypad .\n"); break;
            case XK_KP_Divide:     printf("Keypad /\n"); break;
            case XK_KP_0:          printf("Keypad 0\n"); break;
            case XK_KP_1:          printf("Keypad 1\n"); break;
            case XK_KP_2:          printf("Keypad 2\n"); break;
            case XK_KP_3:          printf("Keypad 3\n"); break;
            case XK_KP_4:          printf("Keypad 4\n"); break;
            case XK_KP_5:          printf("Keypad 5\n"); break;
            case XK_KP_6:          printf("Keypad 6\n"); break;
            case XK_KP_7:          printf("Keypad 7\n"); break;
            case XK_KP_8:          printf("Keypad 8\n"); break;
            case XK_KP_9:          printf("Keypad 9\n"); break;
            default:;
            }
        }
    return(1);
    }
/*=========================================*/
int teclado_menu_inicial(XKeyEvent *theEvent)
    {
    length=XLookupString(theEvent, buffer_tecla, long_max_buffer, &simbolo_tecla,
                                            &theComposeStatus);
    if((simbolo_tecla>='')&&(simbolo_tecla<='~')&&(length>0))
        {
        if((buffer_tecla[0]=='s')||(buffer_tecla[0]=='S'))  return(0);
        if(buffer_tecla[0]=='1')                            return(1);
        if(buffer_tecla[0]=='2')                            return(2);
        if(buffer_tecla[0]=='3')                            return(3);
        if((buffer_tecla[0]=='a')||(buffer_tecla[0]=='A'))  return(4);
        }
    return(10);
    }
/*=========================================*/
int teclado_menu_modo(XKeyEvent *theEvent)
    {
    length=XLookupString(theEvent, buffer_tecla, long_max_buffer, &simbolo_tecla,
                                            &theComposeStatus);
    if((simbolo_tecla>='')&&(simbolo_tecla<='~')&&(length>0))
        {
        if((buffer_tecla[0]=='c')||(buffer_tecla[0]=='C'))  return(0);
        if((buffer_tecla[0]=='s')||(buffer_tecla[0]=='S'))  return(1);
        if((buffer_tecla[0]=='m')||(buffer_tecla[0]=='M'))return(2);
        if((buffer_tecla[0]=='e')||(buffer_tecla[0]=='E'))  return(3);
        }
    return(10);
    }
/*=========================================*/
int teclado_menu_parametros(XKeyEvent *theEvent)
    {
    length=XLookupString(theEvent, buffer_tecla, long_max_buffer, &simbolo_tecla,
                                            &theComposeStatus);
    if((simbolo_tecla>='')&&(simbolo_tecla<='~')&&(length>0))
        {
        if((buffer_tecla[0]=='o')||(buffer_tecla[0]=='O'))return(1);
        if((buffer_tecla[0]=='c')||(buffer_tecla[0]=='C'))return(0);
        }

    return(10);
    }
/*=========================================*/
int teclado_menu_mostrar(XKeyEvent *theEvent)
    {
    length=XLookupString(theEvent, buffer_tecla, long_max_buffer, &simbolo_tecla,
                                            &theComposeStatus);
    if((simbolo_tecla>='')&&(simbolo_tecla<='~')&&(length>0))
        {
        if((buffer_tecla[0]=='c')||(buffer_tecla[0]=='C'))  return(0);
        if((buffer_tecla[0]=='o')||(buffer_tecla[0]=='O'))  return(1);
        if((buffer_tecla[0]=='a')||(buffer_tecla[0]=='A'))  return(2);
        if(buffer_tecla[0]=='-')                            return(3);
        if(buffer_tecla[0]=='+')                            return(4);
        }

    return(10);
    }
/*=========================================*/
int teclado_menu_grabacion(XKeyEvent *theEvent)
    {
    length=XLookupString(theEvent, buffer_tecla, long_max_buffer, &simbolo_tecla,
                                            &theComposeStatus);
    if((simbolo_tecla>='')&&(simbolo_tecla<='~')&&(length>0))
        {
        if((buffer_tecla[0]=='c')||(buffer_tecla[0]=='C'))  return(0);
        if((buffer_tecla[0]=='s')||(buffer_tecla[0]=='S'))  return(1);
        if((buffer_tecla[0]=='n')||(buffer_tecla[0]=='N'))  return(2);
        if((buffer_tecla[0]=='e')||(buffer_tecla[0]=='E'))  return(3);
        }
```

```
    return(10);
    }
/*=========================================*/
int teclado_botonOK(XKeyEvent *theEvent)
    {
    length=XLookupString(theEvent, buffer_tecla, long_max_buffer, &simbolo_tecla,
                                        &theComposeStatus);
    if((simbolo_tecla='')&&(simbolo_tecla<='~')&&(length0))
        { if((buffer_tecla[0]=='o')||(buffer_tecla[0]=='O')) return(0); }

    return(10);
    }
```

textx.c

```
/* textx.c */ /* TEXTOS */
/*=========================================*/
#include<X11/Xlib.h>
#include<string.h>

#include "windows.h"
/*=========================================*/
extern Display *el_display;
/*=========================================*/
XFontStruct *initFont(GC el_GC, char fontName[])
    {
    XFontStruct *fontStruct;
    fontStruct = XLoadQueryFont(el_display, fontName);
    if(fontStruct != 0)
        { XSetFont(el_display, el_GC, fontStruct->fid); }
    return(fontStruct);
    }
/*=========================================*/
void drawImageString(Window theWindow, GC el_GC, int x, int y, char string[])/*dibuja bits*/
    {
    int lenght = strlen(string);
    XdrawImageString(el_display, theWindow, el_GC, x, y, string, lenght);
    }
/*=========================================*/
```

```
void drawString(Window theWindow, GC el_GC, int x, int y, char string[])/*dibuja bits*/
    {
    int lenght = strlen(string);
    XdrawString(el_display, theWindow, el_GC, x, y, string, lenght);
    }
```

windowx.c

```
/* windowx.c */ /* CODIGO PARA ABRIR VENTANAS */
/*=========================================*/
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>

#include "theIcon.h"
#include "windows.h"
/*=========================================*/
extern Display          *el_display;
extern int              el_screen;
extern int              profundidad;
extern unsigned long    pixel_negro;
extern unsigned long    pixel_blanco;
extern Cursor           el_cursor;

#define ANCHO_BORDE 2
/*=========================================*/
int crea_GC(Window nueva_ventana, GC *nuevo_GC)
                    /* crea un contexto grafico para la ventana dada. El contexto grafico */
                    /* es necesario para dibujar .Devuelve 0 si error,  1 si OK . */
    {
    XGCValues  valores_GC;
    *nuevo_GC = XcreateGC( el_display, nueva_ventana, (unsigned long) 0,  &valores_GC);

    if(*nuevo_GC==0)                /* Incapaz de crear un GC */
        { return(0); }
    else                            /* Establece back.y foreg.por defecto en la GC : */
        {
        XSetForeground(el_display, *nuevo_GC, pixel_negro);
        XSetBackground(el_display, *nuevo_GC, pixel_blanco);
```

```
        return(1);
        }
    }
/*=============================================*/
Window abroVentana(int x, int y, int width, int height, int flag, char titulo[], int estado_icono,
                                        Window el_padre, GC *nuevo_GC)
/* int  x, y;          coord. esquina superior izq. de la ventana  */
/* int  width, height; ancho y alto de la ventana                  */
/* int  flag;          si flag > 0,  entonces la ventana es pop-up */
/* char titulo[];      Ttulo para la ventana                       */
/* int  estado_icono;      =1 si empieza como icono,  =0 si no     */
/* Window el_padre; Ventana padre de la nueva ventana              */
/* GC   *nuevo_GC; Contexto Gr<160fico                             */
    {
    XSetWindowAttributes  theWindowAttributes;
    unsigned long         mascara_ventana;
    Window                nueva_ventana;
    XSizeHints            theSizeHints;
    Pixmap                theIconPixmap;
    XWMHints              window_manag_hints;
    XClassHint            hint_clase;

                    /* Establezco los atributos deseados para la ventana : */
    theWindowAttributes.border_pixel     = pixel_negro;
    theWindowAttributes.background_pixel = pixel_blanco;

    if(flag==VENTANA_POP_UP)
        {
        theWindowAttributes.override_redirect=True;
        theWindowAttributes.save_under=True;
        mascara_ventana=(CWBackPixel | CWBorderPixel|CWSaveUnder | CWOverrideRedirect);
        }
    else {
        mascara_ventana=CWBackPixel | CWBorderPixel;
        }
                                    /* abre una ventana sobre el display : */
    nueva_ventana = XCreateWindow( el_display, el_padre, x, y, width, height, ANCHO_BORDE,
                            profundidad, InputOutput, CopyFromParent, mascara_ventana,
                            &theWindowAttributes );

                    /* Establezco  un icono para la ventana.Cada ventana debera*/
                    /* registrar un icono con el window manager. Este icono es  */
                    /* usado si la ventana es reducida a una forma icnica por  */
                    /* el usuario (mediante interaccin con el window manager): */
```



```
theIconPixmap = XCreateBitmapFromData (el_display, nueva_ventana, theIcon_bits,
                        theIcon_width, theIcon_height);
                /* Envo indicaciones ("hints") al Window Manager. Antes de */
                /* que esta ventana aparezca, un X window managr puede*/
                /* interceptar la llamada y poner la ventana donde quiera .*/
                /* Ahora damos al window manager indicaciones (hints) para */
                /* que ponga la ventana donde debe ir :     */
if(estado_icono==0)
    { window_manag_hints.initial_state=NormalState; }
else
    { window_manag_hints.initial_state=estado_icono; }
window_manag_hints.icon_pixmap  = theIconPixmap;
window_manag_hints.flags = IconPixmapHint | StateHint;
XSetWMHints( el_display, nueva_ventana, &window_manag_hints );

        /* almaceno el nombre de la ventana : */
XStoreName(el_display, nueva_ventana, titulo);
XSetIconName(el_display, nueva_ventana, RES_NAME);

hint_clase.res_name  = RES_NAME;
hint_clase.res_class = RES_CLASS;

XSetClassHint(el_display, nueva_ventana, &CopyFromParent hint_clase);

        /* Le digo al window manager el tamaño y localizacion para */
        /* nuestras ventanas. USPosition significa que estamos*/
        /* diciendo al usuario que elija la posicion y tamaño : */
theSizeHints.flags = USPosition | USSize | PMinSize | PMaxSize;
theSizeHints.x = x;
theSizeHints.y = y;
theSizeHints.width = width;
theSizeHints.height = height;
theSizeHints.min_width = 100;
theSizeHints.max_width = 900;
theSizeHints.min_height = 100;
theSizeHints.max_height = 900;

XSetNormalHints(el_display, nueva_ventana, &theSizeHints);

        /* creamos un contexto gráfico para la ventana : */
if(crea_GC(nueva_ventana,nuevo_GC)==0)
    { XDestroyWindow(el_display, nueva_ventana); return( (Window) 0 ); }
            /* X emplaza la ventana visiblemente sobre la screen.*/
            /* Hasta ahora la ventana ha sido creada, pero no aparece */
```

```
        /* sobre la screen. Con XMapWindow lo haremos : */
XMapWindow(el_display, nueva_ventana);

        /* Limpiamos todos los requerimientos al servidor X : */
XFlush(el_display);

        /* Devuelvo el ID de la ventana : */
return(nueva_ventana);
}
```